

[illegible][illegible]

[illegible][illegible]

(2)	116	External and local symbol definitions
(7)	365	Standard tables
(9)	452	P2 buffer verification tables
(10)	548	UNIT_INIT, Unit initialization routine
(11)	575	CLR_IRP - Initialize IRP fields
(12)	600	XMT_FDT, Transmit I/O Operation FDT Routine
(13)	657	RCV_FDT, Read I/O Operation FDT Routine
(16)	725	ALT_START, Alternate Start I/O Routine
(17)	765	XMT_START, Start Transmit Operation
(18)	825	RCV_START, Start Receive Operation
(19)	861	SETMODE_FDT, Set mode I/O operation FDT routine
(20)	963	Complete QIO request routines
(21)	1009	NEW_TRIB - Allocate and init new CDB
(22)	1112	SETMODE_CTRL, Perform setmode FDT operation on controller
(23)	1219	SENSEMODE_FDT, Sense Mode I/O operation FDT routine
(24)	1321	GET_CHAR_RBUF, Get P2 characteristics buffer for read
(24)	1322	GET_CHAR_WBUF, Get P2 characteristics buffer for write
(25)	1369	START_TRIB, Start tributary routine
(26)	1506	LIS_FORK, Listen action routine
(29)	1663	CANCEL, Cancel I/O routine
(30)	1701	CAN_DEV, Device shutdown routine
(31)	1748	ZAP_CDB, Shutdown the tributary
(32)	1880	MSG_FORK, Fork process for receipt of Seq Messages
(33)	1902	DG_FORK, Fork process for receipt of DG
(34)	1997	FINISH_RCV_IO, Finish receive I/O processing
(35)	2048	FILLRCVLIST, Fill receive buffer list
(35)	2049	ADDRCVLIST, Move IRP buffer to free list
(36)	2106	XLATE, Translate Channel to CDB address
(37)	2153	VALIDATE_P2, Validate P2 buffer parameters
(38)	2232	UNPACK_P2_BUF, Unpack a P2 parameter from P2 buffer
(39)	2273	CN_END, End of driver


```
0000 1 .TITLE CNDRIVER - VAX/VMS DECnet-CI Class Driver
0000 2 .IDENT 'V04-000'
0000 3
0000 4 *****
0000 5
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23
0000 24 *****
0000 25
0000 26 ++
0000 27 FACILITY:
0000 28
0000 29 VAX/VMS DECnet-CI class driver
0000 30
0000 31 ABSTRACT:
0000 32
0000 33 This module contains the DECnet-CI class driver FDT routines,
0000 34 SCS dispatcher, and fork routines.
0000 35
0000 36
0000 37
0000 38 Kerbey T. Altmann, 17-Aug-1981
0000 39
0000 40 MODIFIED BY:
0000 41
0000 42 V03-016 ADE3004 A. Eldridge 24-Jul-1984
0000 43 Change name back to DECNET$PHASE_III for now. The change in the
0000 44 name must be phased in by updating the receiver to accept the
0000 45 old and new name before the transmitter can be updated to send
0000 46 the new name. The new name should be DECNET$CI rather than
0000 47 DECNET$PHASE_IV.
0000 48
0000 49 V03-015 LMP0275 L. Mark Pilant, 12-Jul-1984 12:26
0000 50 Initialize the ACL info in the ORB to be a null descriptor
0000 51 list rather than an empty queue. This avoids the overhead
0000 52 of locking and unlocking the ACL mutex, only to find out
0000 53 that the ACL was empty.
0000 54
0000 55 V03-014 LMP0221 L. Mark Pilant, 26-Mar-1984 16:43
0000 56 Change UCB$$_OWNUIC to ORB$$_OWNER.
0000 57
```

```
0000 58 : V03-013 TMK0002 Todd M. Katz 24-Mar-1984
0000 59 : When connecting to a remote station over a specific local
0000 60 : port, which is what this DECnet class driver does, the name of
0000 61 : the local port together with the remote station address must be
0000 62 : specified as an argument to the CONNECT fork process call. This
0000 63 : DECnet class driver was specifying the name of the local port
0000 64 : as PAA. It should now be specifying the name of the local port
0000 65 : as PAA0. If this is not done, the CONNECT will fail.
0000 66 :
0000 67 : Change the SCS process name of the DECnet SYSAP from
0000 68 : DECNET$PHASE_III to DECNET$PHASE_IV.
0000 69 :
0000 70 : V03-012 TMK0001 Todd M. Katz 08-Feb-1984
0000 71 : Use the macro SEND_DG_BUF_REG to do transmits instead of
0000 72 : SEND_DG_BUF. This allows me to remove the pseudo-CDRP which
0000 73 : is currently buried within the CDB. This false CDRP was only
0000 74 : being used to pass the application data and CDT addresses to
0000 75 : the fork process call, FPC$SENDDG. The fork process call issued
0000 76 : by SEND_DG_BUF_REG, FPC$SENDRGDG, requires these addresses to be
0000 77 : in registers when it is invoked, and thus, doesn't require a
0000 78 : CDRP in order to obtain them.
0000 79 :
0000 80 : V03-011 ADE3003 Alan D. Eldridge 19-May-1983
0000 81 : Replaced constants with appropriate SBO$ symbols.
0000 82 :
0000 83 : V03-010 ADE3002 Alan D. Eldridge 19-Apr-1983
0000 84 : Modified datagram internal SCS header "size" field to
0000 85 : handle new negative offset processing option.
0000 86 :
0000 87 : V03-009 ADE3001 Alan D. Eldridge 2-Feb-1983
0000 88 : Simplified connect/disconnect control. Removed the sending of
0000 89 : XON/XOFF sequenced messages. Issue a CONNECT only if the
0000 90 : remote sequence number is higher. Redefined the CDB.
0000 91 :
0000 92 : V03-008 NPK3010 N. Kronenberg 24-Nov-1982
0000 93 : Removed output array specifier from CONFIG_SYS call.
0000 94 :
0000 95 : V03-007 KTA0109 Kerbey T. Altmann 8-Jul-1982
0000 96 : Fix bug in returning buffer info in SENSEMODE.
0000 97 :
0000 98 : V03-006 NPK3004 N. Kronenberg 2-Jul-1982
0000 99 : Modify START_TRIB to connect over specific virtual
0000 100 : circuit instead of looking up the remote system and
0000 101 : connecting to that.
0000 102 :
0000 103 : V03-005 NPK3003 N. Kronenberg 1-Jul-1982
0000 104 : Fixed offsets from CONFIG_PTH/SYS for new format
0000 105 : returned by those routines.
0000 106 :
0000 107 : V03-004 KDM0002 Kathleen D. Morse 28-Jun-1982
0000 108 : Added $PRDEF.
0000 109 :
0000 110 : V03-003 KTA0097 Kerbey T. Altmann 20-Apr-1982
0000 111 : Fix bad branch destination.
0000 112 :
0000 113 :
0000 114 :--
```



```
0000 116      .SBTTL External and local symbol definitions
0000 117
0000 118      :
0000 119      : System definitions
0000 120      :
0000 121
0000 122      $ADPDEF      : Adapter control block
0000 123      $CDTDEF      : Connection descriptor
0000 124      $CRBDEF      : Channel request block
0000 125      $CXBDEF      : Complex buffers
0000 126      $DCDEF      : Device classes and types
0000 127      $DDBDEF      : Device data block
0000 128      $DEVDEF      : Device characteristics
0000 129      $DPTDEF      : Driver prologue table defs
0000 130      $DYNDEF      : Control block defs
0000 131      $IODEF      : I/O function codes
0000 132      $IPLDEF      : Hardware IPL definitions
0000 133      $JIBDEF      : Job info block
0000 134      $IRPDEF      : I/O request packet
0000 135      $NMADEF      : Network Management definitions
0000 136      $ORBDEF      : Object's Rights Block
0000 137      $PBDDEF      : Path block definitions
0000 138      $PCBDEF      : Process control block
0000 139      $PDTDEF      : Port Descriptor Table
0000 140      $PRDEF      : Processor register definitions
0000 141      $SBDEF      : System block definitions
0000 142      $SBODEF      : System block output definitions
0000 143      $SSDEF      : System status codes
0000 144      $UCBDEF      : Unit control block
0000 145      $VECDDEF      : Interrupt vector block
0000 146      $XMDEF      : XMDRIVER symbols
0000 147
0000 148      :
0000 149      : Local macros
0000 150      :
0000 151      .MACRO SETBIT POS,BAS,?L      : Set a single bit
0000 152      BBSS POS,BAS,L
0000 153      L:
0000 154      .ENDM SETBIT
0000 155
0000 156      .MACRO CLRBIT POS,BAS,?L      : Clear a single bit
0000 157      BBCC POS,BAS,L
0000 158      L:
0000 159      .ENDM CLRBIT
0000 160
0000 161      .MACRO PUSHQ ARG      : Push a quadword
0000 162      MOVQ ARG, -(SP)      : Save argument on stack
0000 163      .ENDM PUSHQ
0000 164
0000 165      .MACRO POPQ ARG      : Pop a quadword
0000 166      MOVQ (SP)+, ARG      : Restore argument
0000 167      .ENDM POPQ
0000 168
```

```
0000 170
0000 171 $DEFINI PARAM
0000 172
00000004 0000 173 COUNT_C_ENTRY = 2*2 ; COUNT table entry size
0000000C 0000 174 PARAM_C_ENTRY = 2*6 ; PARAM table entry size
0000 175 _VIELD PRM,0,<- ; Parameter bits and sizes
0000 176 <TYPE,12,M>,- ; Parameter type
0000 177 <MIN,1,M>,- ; Parameter minimum value
0000 178 <MAX,1,M>,- ; Parameter maximum value
0000 179 <REQUIRED,1,M>,- ; Parameter required flags
0000 180 <INVALID,1,M>,- ; Parameter invalid flags
0000 181 >
0000 182
0000 183 _VIELD OFF,0,<- ; Offset word fields
0000 184 <VALUE,10,M>,- ; Offset value
0000 185 <WIDTH,6,M>,- ; Width of field in structure
0000 186 >
0000 187
0000 188 $DEFEND PARAM
0000 189
0000 190 .MACRO PARAM TYPE,OFFSET,WIDTH=0,MIN=0,MAX=-1,REQUIRED,INVALID
0000 191 ; NOTE - The REQUIRED field can only check 1 bit!
0000 192
0000 193 $$$NUM = $$$NUM+1 ; Count number of time executed
0000 194 $$$TYP = NMASC 'TYPE & PRM_M_TYPE ; Isolate type
0000 195 $$$OFF = OFFSET & OFF_M_VALUE ; Isolate offset only
0000 196
0000 197 .IIF NOT_BLANK <MIN>, $$$TYP = $$$TYP ! PRM_M_MIN
0000 198 .IIF NOT_BLANK <MAX>, $$$TYP = $$$TYP ! PRM_M_MAX
0000 199 .IIF NOT_BLANK <REQUIRED>, $$$TYP = $$$TYP ! PRM_M_REQUIRED
0000 200 .IIF NOT_BLANK <INVALID>, $$$TYP = $$$TYP ! PRM_M_INVALID
0000 201
0000 202 .WORD $$$TYP
0000 203 .WORD $$$OFF ! <WIDTH @ OFF_V_WIDTH>
0000 204 .WORD MIN
0000 205 .WORD MAX
0000 206 .WORD REQUIRED
0000 207 .WORD INVALID
0000 208 .ENDM PARAM
0000 209
0000 210
0000 211 .MACRO COUNT TYPE,OFFSET,WIDTH=32
0000 212
0000 213 $$$NUM = $$$NUM+1 ; Bump number of time executed
0000 214 $$$OFF = OFF_M_VALUE & OFFSET ; Isolate offset only
0000 215 $$$TYP = PRM_M_TYPE & NMASC 'TYPE ; Isolate type
0000 216
0000 217 .IIF IDN, WIDTH, 8, $$$TYP = $$$TYP ! <1@NMASV_CNT_WID>
0000 218 .IIF IDN, WIDTH, 16, $$$TYP = $$$TYP ! <2@NMASV_CNT_WID>
0000 219 .IIF IDN, WIDTH, 32, $$$TYP = $$$TYP ! <3@NMASV_CNT_WID>
0000 220
0000 221
0000 222 .WORD $$$TYP ! NMASM_CNT_COU
0000 223 .WORD $$$OFF ! <WIDTH @ OFF_V_WIDTH>
0000 224 .ENDM COUNT
0000 225
0000 226 .MACRO START_TABLE NAME ; Start Table declaration
```

```
0000 227 $$$NUM = 0 ; Init number of entries
0000 228 'NAME' TABLE = . ; Define beginning of table
0000 229 .ENDM START_TABLE
0000 230
0000 231 .MACRO END TABLE NAME ; Terminate Table declaration
0000 232 ;WORD 0 ; Create marker
0000 233 'NAME' NUM = $$$NUM ; Number of entries
0000 234 .ENDM END_TABLE
0000 235
0000 236 ;
0000 237 ; Local symbols
0000 238 ;
0000 239 ;
0000 240 ;
0000 241 ; $QIO parameter offsets
0000 242 ;
00000000 0000 243 P1 = 0 ; Parameter 1
00000004 0000 244 P2 = 1*4 ; Parameter 2
0000 245 ;
0000 246 ;
0000 247 ; Other constants
0000 248 ;
00000009 0000 249 RBFMIN = 9 ; Min size of CDB_B_RCV_CNT
0000001F 0000 250 RBFMAX = 31 ; Max size of CDB_B_RCV_CNT
00000006 0000 251 RBFTHR = 6 ; CND_B_RCV_FQ threshold. Below this
0000 252 ; signal XMSM_STS_BUFFAIL in IOST2
00000010 0000 253 MAX_TRB = 16 ; Max tributaries on CI device
```



```
0000 255 :  
0000 256 : Overlays of IRP  
0000 257 :  
0000 258 ASSUME IRP$L_SEGVBN EQ IRP$Q_NT_PRVMSK+8  
0000 259  
0000 260 $DEFINI IRP  
0000 261  
00000040 0000 262 = IRP$Q_NT_PRVMSK ; Overlay network priv mask  
0040 263 $DEF IRP$B_INDEX .BLKB 1 ; Vector index for CDB  
0041 264  
00000054 0041 265 = IRP$L_EXTEND  
0054 266 $DEF IRP$L_CDB .BLKL 1  
0058 267  
0058 268 $DEFEND IRP ; End of IRP overlays  
0000 269  
0000 270  
0000 271 :  
0000 272 : Definitions that follow the standard UCB fields  
0000 273 :  
0000 274  
0000 275 $DEFINI UCB ; Start of UCB definitions  
0000 276  
00000090 0000 277 . = UCB$C_LENGTH ; Position at end of UCB  
0090 278  
0090 279 $DEF UCB$L_LIS_CDT .BLKL 1 ; Addr of listening CDT  
0094 280 $DEF UCB$L_TWIR_CDT .BLKL 1 ; Addr of loopbacked accept CDT  
0098 281 $DEF UCB$L_DGHDRSZ .BLKL 1 ; Size of the SCS header for DG's  
009C 282 $DEF UCB$W_DUMMY .BLKW 1 ; Dummy location for unwanted param's  
009E 283 $DEF UCB$B_CN_PORT .BLKB 1 ; Our port number  
009F 284 $DEF UCB$B_RCV_CNT .BLKB 1 ; Number of receive buffers  
00A0 285 $DEF UCB$L_VEC_CDB .BLKL MAX_TRB ; CDB address vector  
00E0 286 $DEF UCB$W_VEC_CHAN .BLKW MAX_TRB ; User channel lookup vector  
0100 287  
00000100 0100 288 UCB$C_CN_LENGTH = <.+15>&-16 ; Size of UCB padded to a quadword  
0100 289  
0100 290 :  
0100 291 : Define device status bits  
0100 292 :  
0100 293 $VIELD UCB,0,<- ; CNDRIVER UCB$W_DEVSTS bits  
0100 294 <CN_INITED,,M>,- ; Device init'ed  
0100 295 >  
0100 296 $DEFEND UCB ; End of UCB definitions  
0000 297
```

```
0000 299 : CNDRIVER CDB definitions
0000 300 :
0000 301 :
0000 302 $DEFINI CDB
0000 303 :
0000 304 $DEF CDB_Q_FORK .BLKQ 1 : Fork Queue Linkage
0008 305 $DEF CDB_W_SIZE .BLKW 1 : Structure size
000A 306 $DEF CDB_B_TYPE .BLKB 1 : Structure type
000B 307 $DEF CDB_B_FIPL .BLKB 1 : Fork IPL (not UCB FIPL)
00000006 000C 308 CDB_C_FIPL = 6 : Must be less than SCS's IPL (8)
000C 309 $DEF CDB_L_FPC .BLKL 1 : Fork PC
0010 310 $DEF CDB_L_FR3 .BLKL 1 : Fork R3
0014 311 $DEF CDB_L_FR4 .BLKL 1 : Fork R4
0018 312 :
0018 313 $DEF CDB_Q_XMT_IRP .BLKQ 1 : Transmit IRP's awaiting completion
0020 314 $DEF CDB_Q_RCV_IRP .BLKQ 1 : Receive IRP's awaiting buffers
0028 315 $DEF CDB_Q_RCV_MSG .BLKQ 1 : Receive buffers containing messages
0030 316 :
0030 317 $DEF CDB_L_SETMODE .BLKL 1 : Ptr to IOS SETMODE
0034 318 $DEF CDB_L_ABSTIME .BLKL 1 : Time last DISCONNECT was issued
0038 319 $DEF CDB_W_BUFSIZ .BLKW 1 : Buffer size
003A 320 $DEF CDB_W_STS .BLKW 1 : Circuit status
003C 321 $DEF CDB_B_RCV_CNT .BLKB 1 : Receive buffer count
003D 322 $DEF CDB_B_RCV_FQ .BLKB 1 : Receive buffers on free queue
003E 323 $DEF CDB_B_TRB_ADDR .BLKB 1 : Tributary address
003F 324 $DEF CDB_B_STA .BLKB 1 : Circuit state
0040 325 :
0040 326 :
0040 327 : Circuit counters
0040 328 :
0040 329 :
0040 330 $DEF CDB_L_BRC .BLKL 1 : Receive byte count
0044 331 $DEF CDB_L_BSN .BLKL 1 : Transmit byte count
0048 332 $DEF CDB_L_DBR .BLKL 1 : Data buffers received
004C 333 $DEF CDB_L_DBS .BLKL 1 : Data buffers sent
0050 334 :
0050 335 $DEF CDB_L_UCB .BLKL 1 : Addr of UCB
0054 336 $DEF CDB_L_CDT .BLKL 1 : Ptr to CDT
0058 337 $DEF CDB_B_REMVER .BLKB 1 : Remote's protocol version
0059 338 $DEF CDB_B_REMSYS .BLKB 1 : Remote's operating system
00000058 005A 339 CDB_W_REMPROT = CDB_B_REMVER : Label combining two fields above
005A 340 $DEF CDB_B_DUMMY .BLKB 1 : Dummy location for unwanted param's
005B 341 $DEF CDB_B_RSTCNT .BLKB 1 : Restart counter for slowing down
005C 342 : restart frequency
00000060 005C 343 CDB_C_LENGTH = <.+15>&-16 : Pad structure out to a quadword
005C 344 :
005C 345 :
005C 346 : Define status bits used in CDB_W_STS and CDB_B_STA values
005C 347 :
005C 348 : _VIELD CDB,0,<- : Tributary status bits for CDB_W_STS
005C 349 : <RUN,,M>,- : Tributary is in RUN state
005C 350 : <CONN,,M>,- : Call to CONNECT pending
005C 351 : <ACPT,,M>,- : CALL to ACCEPT pending
005C 352 : <DISC,M>,- : Call to DISCONNECT or FORK pending
005C 353 : <REJECT,,M>,- : Call to REJECT pending
005C 354 :
005C 355 :
```

CNDRIVER
V04-000

- VAX/VMS DECnet-CI Class Driver I 10
External and local symbol definitions

16-SEP-1984 01:19:27 VAX/VMS Macro V04-00
5-SEP-1984 00:11:06 [DRIVER.SRC]CNDRIVER.MAR;1

Page 8
(6)

00000000	005C	356	CDB_C_IDLE = 0
00000001	005C	357	CDB_C_OPEN = 1
00000002	005C	358	CDB_C_CONN = 2
00000003	005C	359	CDB_C_LSTN = 3
00000004	005C	360	CDB_C_ACPT = 4
	005C	361	
	005C	362	\$DEFEND CDB
	0000	363	

: Tributary is idle
: Tributary connection has been made
: Tributary has CONNECT pending
: Tributary is listening for connect
: Tributary has ACCEPT pending


```
0000 365      .SBTTL Standard tables
0000 366
0000 367
0000 368      : Driver prologue table
0000 369      :
0000 370
0000 371      DPTAB      -
0000 372      END      = CN_END,-
0000 373      ADAPTER  = NUCL,-
0000 374      FLAGS   = DPTSM_SCS,-
0000 375      UCBSIZE = UCBS_CN_LENGTH,-
0000 376      NAME    = CNDRIVER,-
0000 377
0038 378      DPT_STORE INIT
0038 379
0038 380      DPT_STORE UCB,UCBSB_FIPL,B,8
003C 381      DPT_STORE UCB,UCBSB_DIPL,B,8
0040 382      DPT_STORE ORB,ORBSB_FLAGS,B,-
0040 383      ZORBSM_PROT_16>
0044 384      DPT_STORE ORB,ORBSW_PROT,B,0
0049 385      DPT_STORE ORB,ORBSL_OWNER,L,<^X010001>
0050 386      DPT_STORE UCB,UCBSL_DEVCHAR,L,-
0050 387      ZDEVSM_NET!-
0050 388      DEVSM_REC!-
0050 389      DEVSM_IDV!-
0050 390      DEVSM_ODV-
0050 391      >
0057 392      DPT_STORE UCB,UCBSB_DEVCLASS,B,DCS_SCOM
0058 393      DPT_STORE UCB,UCBSW_DEVBUSIZ,B,W,-
0058 394      SCSSGW_MAXDG
0062 395
0062 396      DPT_STORE REINIT
0062 397
0062 398      DPT_STORE DDB,DDBSL_DDT,D,CNSDDT
0067 399      DPT_STORE CRB,-
0067 400      CRBSL_INTD+VECSL_UNITINIT,-
0067 401      D,UNIT_INIT
006C 402
006C 403      DPT_STORE END
0000 404
                                : DPT-creation macro
                                : End of driver label
                                : Adapter type
                                : Driver requires SCS
                                : Length of UCB
                                : Driver name
                                :
                                : Start of load
                                : initialization table
                                : Device fork IPL
                                : Device interrupt IPL
                                : Protection block flags
                                : SOGW protection word
                                : default protection
                                : [1,1] owns the device
                                : Device characteristics
                                : e.g., network device
                                : record oriented
                                : input device
                                : output device
                                :
                                : Sample device class
                                : Default buffer size
                                :
                                : Start of reload
                                : initialization table
                                : Address of DDT
                                : Address of device
                                : unit initialization
                                : routine
                                :
                                : End of initialization
                                : tables
```

```
0000 406 :  
0000 407 : Driver dispatch table  
0000 408 :  
0000 409 : DDTAB - : DDT-creation macro  
0000 410 : DEVNAM = CN,- : Name of device  
0000 411 : FUNCTB = CN_FUNCTABLE,- : FDT address  
0000 412 : CANCEL = CANCEL,- : Cancel I/O routine  
0000 413 : ALTSTART= ALT_START : Alternate start I/O  
0038 414 :  
0038 415 : Function dispatch table  
0038 416 :  
0038 417 CN_FUNCTABLE:  
0038 418 FUNCTAB ,- : FDT for driver  
0038 419 : Valid I/O functions  
0038 420 : <READBLK,- : Read logical  
0038 421 : WRITELBLK,- : Write logical  
0038 422 : SETMODE,- : Set device mode  
0038 423 : SENSEMODE,- : Sense mode  
0038 424 : SETCHAR - : Set device chars.  
0040 425 :  
0040 426 : FUNCTAB ,- : Buffered functions:  
0040 427 : <READBLK,- : Read logical  
0040 428 : WRITELBLK,- : Write logical  
0040 429 : SETMODE,- : Set device mode  
0040 430 : SENSEMODE,- : Sense mode  
0040 431 : SETCHAR - : Set device chars.  
0048 432 :  
0048 433 : FUNCTAB CLR_IRP,- : Init IRP fields  
0048 434 : <READBLK,- : Read logical  
0048 435 : WRITELBLK,- : Write logical  
0048 436 : SETMODE,- : Set device mode  
0048 437 : SENSEMODE,- : Sense mode  
0048 438 : SETCHAR - : Set device chars.  
0054 439 :  
0054 440 : FUNCTAB RCV_FDT,- : FDT read routine for  
0054 441 : <READBLK,- : read logical,  
0060 442 :  
0060 443 : FUNCTAB XMT_FDT,- : FDT write routine for  
0060 444 : <WRITELBLK,- : write logical,  
006C 445 :  
006C 446 : FUNCTAB SETMODE_FDT,- : FDT set mode routine  
006C 447 : <SETMODE,- : set mode  
006C 448 : SETCHAR - : set characteristics  
0078 449 :  
0078 450 : FUNCTAB SENSEMODE_FDT,- : FDT sense mode routine  
0078 450 : <SENSEMODE> : for sensemode
```

```
0084 452      .SBTTL P2 buffer verification tables
0084 453
0084 454      :: Define CDB parameters
0084 455      ::
0084 456      START_TABLE TRIB_PRM      ; Start of tributary parameter table
0084 457
0084 458      PARAM PCCI_MST, OFFSET = CDB_B_DUMMY,-      ; Trib maint state
0084 459      WIDTH = 0,-      ; Dummy location
0084 460      MIN = NMASC_STATE_ON,-
0084 461      MAX = NMASC_STATE_OFF,-
0084 462      REQUIRED= 0,-
0084 463      INVALID = CDB_M_RUN
0084 464
0090 465      PARAM PCCI_TRI, OFFSET = CDB_B_TRB_ADDR,-      ; Trib address
0090 466      WIDTH = 8,-
0090 467      MIN = 0,-
0090 468      MAX = 15,-
0090 469      REQUIRED= 0,-
0090 470      INVALID = CDB_M_RUN
0090 471
009C 472      PARAM PCCI_MRB, OFFSET = CDB_B_RCV_CNT,-      ; Trib max buf
009C 473      WIDTH = 8,-
009C 474      MIN = 0,-
009C 475      MAX = 255,-
009C 476      REQUIRED= 0,-
009C 477      INVALID = CDB_M_RUN
009C 478
00A8 479      END_TABLE TRIB_PRM      ; End of tributary parameter table
00A8 480
00AA 481      :: Define UCB parameters
00AA 482      ::
00AA 483      START_TABLE LINE_PRM      ; Start of device parameter table
00AA 484
00AA 485      PARAM PCLI_DUP, OFFSET = UCBSW_DUMMY,-      ; Duplex
00AA 486      WIDTH = 0,-      ; Dummy location
00AA 487      MIN = NMASC_DPX_FUL,-
00AA 488      MAX = NMASC_DPX_HAL,-
00AA 489      REQUIRED= 0,-
00AA 490      INVALID = UCBSM_CN_INITED
00AA 491
00B6 492      PARAM PCLI_CON, OFFSET = UCBSW_DUMMY,-      ; Controller mode
00B6 493      WIDTH = 0,-      ; Dummy location
00B6 494      MIN = NMASC_LINCN_NOR,-
00B6 495      MAX = NMASC_LINCN_LOO,-
00B6 496      REQUIRED= 0,-
00B6 497      INVALID = UCBSM_CN_INITED
00B6 498
00C2 499      PARAM PCLI_BUS, OFFSET = UCBSW_DEVBUSIZ,-      ; Block size
00C2 500      WIDTH = 16,-
00C2 501      MIN = 32,-
00C2 502      MAX = 948,-
00C2 503      REQUIRED= 0,-
00C2 504      INVALID = UCBSM_CN_INITED
00C2 505
00CE 506
```



```

00CE 509      PARAM PCLI_BFN, OFFSET = UCBSB_RCV_CNT,-      ; Maximum receive buffers
00CE 510      WIDTH      = 8,-
00CE 511      MIN        = 1,-
00CE 512      MAX        = 255,-
00CE 513      REQUIRED= 0,-
00CE 514      INVALID = UCBSM_CN_INITED
00DA 515
00DA 516 END_TABLE LINE_PRM      ; End of device parameter tables
00DC 517
00DC 518 ;
00DC 519 ; Tributary counter type codes
00DC 520 ;
00DC 521 START_TABLE TRIB_CNT      ; Start of Tributary COUNTER table
00DC 522
00DC 523      COUNT   CTCIR_BRC, WIDTH=32, OFFSET=CDB_L_BRC      ; Bytes received
00E0 524      COUNT   CTCIR_BSN, WIDTH=32, OFFSET=CDB_L_BSN      ; Bytes sent
00E4 525      COUNT   CTCIR_DBR, WIDTH=32, OFFSET=CDB_L_DBR      ; Data blocks received
00E8 526      COUNT   CTCIR_DBS, WIDTH=32, OFFSET=CDB_L_DBS      ; Data blocks sent
00EC 527
00EC 528 END_TABLE      TRIB_CNT      ; End of Tributary COUNTER table
00EE 529
00EE 530
00EE 531 START_TABLE LINE_CNT      ; Start of device COUNTER table
00EE 532 END_TABLE      LINE_CNT      ; - null table
00F0 533
00F0 534 ;
00F0 535 ; Our SCS process name and connect data
00F0 536 ;
00F0 537 PROC_C_NAM = 6      ; How much of PROC_NAM must match
00F0 538 PROC_NAM:
00F0 539      .ASCII 'DECNET$PHASE_III'      ; How SCS knows us -- 16 characters long
00FC 540
0100 540 CONN_DATA:
0100 541      .BYTE 1      ; Protocol version
0101 542      .BYTE 0      ; Operating system (VMS) id
0102 543      .BYTE 0[14] ; Remaining fields must be zero
010E 544
0110 545 OLD_C_PROT = 0      ; Use for original protocol
0110 546

```

```

0110 548 .SBTTL UNIT_INIT, Unit initialization routine
0110 549
0110 550 :++
0110 551 UNIT_INIT - Readies unit for I/O operations
0110 552
0110 553
0110 554 The operating system calls this routine after calling the
0110 555 controller initialization routine:
0110 556
0110 557 - at system startup
0110 558 - during driver loading
0110 559 - during recovery from a power failure
0110 560
0110 561 The unit is put online.
0110 562
0110 563 Inputs: R5 = UCB address
0110 564
0110 565 Outputs: All registers are preserved
0110 566
0110 567
0110 568 :--
0110 569
64 A5 10 A8 0110 570 UNIT_INIT: ; Initialize unit
05 0110 571 BISW #UCBSM_ONLINE,UCBSW_STS(R5) ; Set unit online
0114 572 RSB ; Return
0115 573

```

```

0115 575 .SBTTL CLR_IRP - Initialize IRP fields
0115 576
0115 577 :++
0115 578 CLR_IRP - Initialize IRP fields
0115 579
0115 580
0115 581 Selected IRP fields are initialized. The function code with modifiers
0115 582 is setup.
0115 583
0115 584 Inputs: R3 IRP address
0115 585
0115 586 Outputs: All other registers are preserved.
0115 587
0115 588 IPL may be FIPL or ASTDEL
0115 589
0115 590 :--
0115 591 CLR_IRP:
0115 592 CLRQ IRPSL_IOST1(R3) ; Initialize IRP fields
0118 593 CLRL IRPSL_SVAPTE(R3) ; Clear IOSB image
0118 594 CLRW IRPSW_BOFF(R3) ; Init buffer pointer
011E 595 CLRL IRPSL_CDB(R3) ; No quota to return yet at I/O post
0121 596 CLRQ IRPSB_INDEX(R3) ; No CDB yet
0124 597 RSB ; No trib i.d. yet
0125 598 ; Done

```



```
0125 600 .SBTTL XMT_FDT, Transmit I/O Operation FDT Routine
0125 601
0125 602
0125 603 :++
0125 604 : XMT_FDT - Transmit I/O Operation FDT Routine
0125 605
0125 606 : This routine is called by the SYS$QIO system service to dispatch a
0125 607 : WRITE I/O request. The buffer is validated for access and copied to a
0125 608 : system buffer.
0125 609
0125 610 : The QIO parameters used for WRITES are:
0125 611
0125 612 : P1 = address of the buffer
0125 613 : P2 = size of the buffer
0125 614
0125 615 : Inputs:
0125 616 : R3 - IRP address (I/O request packet)
0125 617 : R4 - PCB address (process control block)
0125 618 : R5 - UCB address (unit control block)
0125 619 : R6 - CCB address (channel control block)
0125 620 : R7 - bit number of the I/O function code
0125 621
0125 622 : IPL = ASTDEL (2)
0125 623
0125 624 : Outputs:
0125 625 : R0 = status of transmit request initiation
0125 626 : R1,R2 are clobbered, all others are preserved.
0125 627 :--
0125 628 XMT_FDT:
0125 629 BSBB XMT_RCV_FDT_CO ; Transmit FDT routine
0125 630 JSB G^EXES$WRITECHK ; Get user buffer
0125 631 ; - no return on error
0125 632 GET_BUF: ; Check buffer access
0125 633 PUSHF #M<R1,R2,R3,R4,R5> ; (no return means no access)
0125 634 ; Get buffer
0125 635 JSB G^EXES$BUFFRQUOTA ; Save registers
0125 636 BLBC R0,20$ ; Check if process has sufficient qu
0125 637 ADDL #CXB$C OVERHEAD,R1 ; If LBC quota check failure
0125 638 JSB G^EXES$ALONONPAGED ; Add in overhead
0125 639 BLBC R0,20$ ; Allocate buffer for output
0125 640 ADDL3 #<DYN$C CXB@16>,R1,IRPSW_SIZE(R2) ; If LBC allocation failure
0125 641 MOVAB CXB$C HEADER(R2),(R2) ; Set the size
0125 642 MOVL (SP),R1 ; Store pointer to data area
0125 643 MOVL R2,4(SP) ; Get back message size
0125 644 MOVL 8(SP),R3 ; Save buffer address
0125 645 MOVL PCB$C JIB(R4),R0 ; Retrieve address of IRP
0125 646 SUBW R1,JIB$C BYTCNT(R0) ; Get JIB address
0125 647 MOVL R2,IRPSW_SIZE(R3) ; Adjust buffered I/O quota
0125 648 MOVW R1,IRPSW_BOFF(R3) ; Setup buffer pointer
0125 649 BEQL 10$ ; Set number of bytes charged to quo
0125 650 BBS #IRPSW_FUNC,IRPSW_STS(R3),10$ ; If EQL then none
0125 651 MOVC3 R1,@IRPSW_IOST2(R3),@R2 ; If BS then "read" function
0125 652 10$: MOVL #1,R0 ; Move data
0125 653 ; Indicate success
0125 654 20$: POPR #M<R1,R2,R3,R4,R5> ; Restore registers
0125 655 RSB ; Return to co-routine with
```

69 10 0125 628 BSBB XMT_RCV_FDT_CO ; Transmit FDT routine
00000000'GF 16 0127 629 JSB G^EXES\$WRITECHK ; Get user buffer
3E BB 012D 632 GET_BUF: ; Check buffer access
00000000'GF 16 012F 633 PUSHF #M<R1,R2,R3,R4,R5> ; (no return means no access)
49 50 E9 0135 634 ; Get buffer
51 0000004C 8F C0 0138 635 JSB G^EXES\$BUFFRQUOTA ; Save registers
00000000'GF 16 013F 636 BLBC R0,20\$; Check if process has sufficient qu
39 50 E9 0145 637 ADDL #CXB\$C OVERHEAD,R1 ; If LBC quota check failure
0B A2 51 001B0000 8F C1 0148 638 JSB G^EXES\$ALONONPAGED ; Add in overhead
62 48 A2 9E 0151 639 BLBC R0,20\$; Allocate buffer for output
51 6E DO 0155 640 ADDL3 #<DYN\$C CXB@16>,R1,IRPSW_SIZE(R2) ; If LBC allocation failure
04 AE 52 DO 0158 641 MOVAB CXB\$C HEADER(R2),(R2) ; Set the size
53 08 AE DO 015C 642 MOVL (SP),R1 ; Store pointer to data area
50 0080 C4 DO 0160 643 MOVL R2,4(SP) ; Get back message size
20 A0 51 A2 0165 644 MOVL 8(SP),R3 ; Save buffer address
2C A3 52 DO 0169 645 MOVL PCB\$C JIB(R4),R0 ; Retrieve address of IRP
30 A3 51 B0 016D 646 SUBW R1,JIB\$C BYTCNT(R0) ; Get JIB address
06 2A A3 01 E0 0173 647 MOVL R2,IRPSW_SIZE(R3) ; Adjust buffered I/O quota
00 B2 3C B3 51 28 0178 648 MOVW R1,IRPSW_BOFF(R3) ; Setup buffer pointer
50 01 DO 017E 649 BEQL 10\$; Set number of bytes charged to quo
3E BA 0181 650 BBS #IRPSW_FUNC,IRPSW_STS(R3),10\$; If EQL then none
05 0183 651 MOVC3 R1,@IRPSW_IOST2(R3),@R2 ; If BS then "read" function
10\$: MOVL #1,R0 ; Move data
20\$: POPR #M<R1,R2,R3,R4,R5> ; Restore registers
RSB ; Return to co-routine with

```

0184 657 .SBTTL RCV_FDT, Read I/O Operation FDT Routine
0184 658
0184 659 :++
0184 660 : RCV_FDT - Read I/O Operation FDT Routine
0184 661 :
0184 662 :
0184 663 : This routine is called by the SYS$QIO system service to dispatch a
0184 664 : READ I/O request.
0184 665 :
0184 666 : The QIO parameters for READs are:
0184 667 :
0184 668 :     P1 = address of the buffer
0184 669 :     P2 = size of the buffer
0184 670 :     All other parameters are unused.
0184 671 :
0184 672 : The specified buffer is checked for accessibility. The buffer address and
0184 673 : count are saved in the packet. Then IPL is raised to device fork IPL and if
0184 674 : a message is available the operation is complete. Otherwise the packet is
0184 675 : queued onto the waiting receive list of the CDB.
0184 676 :
0184 677 :
0184 678 : Inputs:      R3 - IRP address (I/O request packet)
0184 679 :             R4 - PCB address (process control block)
0184 680 :             R5 - UCB address (unit control block)
0184 681 :             R6 - CCB address (channel control block)
0184 682 :             R7 - bit number of the I/O function code
0184 683 :
0184 684 :             IPL = ASTDEL (2)
0184 685 :
0184 686 : Outputs:     R0 = status of transmit request initiation
0184 687 :
0184 688 :             R1,R2 are clobbered, all others are preserved.
0184 689 :
0184 690 :--
0184 691 RCV_FDT:
0184 692 BSBB  XMT_RCV_FDT_CO      : Read FDT process routine
0186 693 JSB  G*EXE$READCHK     : Get user buffer
0186 694 JSB  G*EXE$READCHK     : - no return on error
018C 695 MOVL #1,R0            : Check accessibility
018C 696 RSB                  : (No return on no access)
018F 697                    : Say "success"
0190 698                    : Return status to co-routine

```

```

      OA  10
00000000'GF  16
      50  01  D0
              05

```

```
0190 700
0190 701 XMT_RCV_FDT_CO:
0190 702 MOVZWL S^#SS$_BADPARAM,R0 ; Assume bad parameters
0193 703 MOVZWL P2(AP),R1 ; Get buffer size
0197 704 BEQL 10$ ; If zero, abort I/O request
0199 705 CMPW R1,UCB$W_DEVBUSIZ(R5) ; Is buffer too big?
019D 706 BGTRU 10$ ; If GTRU yes, abort I/O request
019F 707 MOVL P1(AP),R0 ; Get user buffer virt address
01A2 708 MOVL R0,IRP$L_IOST2(R3) ; Save it for MOVC
01A6 709 JSB @($P)+ ; Call back our caller
01A8 710 BLBS R0,20$ ; If LBS, continue
01AB 711 BRW ABORT_REQ ; Abort the request
01AE 712 10$:
01AE 713 20$:
01AE 714
01AE 715
01AE 716
01AE 717
01AE 718
00000000'GF 9F 01AE 719 PUSHAB G^EXESQIORETURN ; Setup return address on stack
01B4 720 SETIPL UCB$B_FIPL(R5) ; Raise IPL to fork level
01B8 721 ; to lock the data base
01B8 722
01B8 723 Fall thru to ALT_START
```

Okay so far. Setup to return to EXESQIORETURN -- which returns to the user with SS\$NORMAL in R0. This means that all subsequent errors must be reported via the IOSB.


```
01B8 725 .SBTTL ALT_START, Alternate Start I/O Routine
01B8 726
01B8 727 :++
01B8 728 : ALT_START - Alternate Start I/O Routine
01B8 729
01B8 730 :
01B8 731 : This entry point is used to dispatch IOS_READBLK and IOS_WRITEBLK requests.
01B8 732 : The IRP is either built by our own FDI routines, or by some higher level
01B8 733 : Executive agent (e.g., NETDRIVER). All I/O status, including errors, must
01B8 734 : be passed via IOPOST in the IOSB.
01B8 735 :
01B8 736 :
01B8 737 : NOTE: The CHAN field of the IRP is sufficient to map to a CDB.
01B8 738 :
01B8 739 : Inputs: R3 - IRP address
01B8 740 : R5 - UCB address
01B8 741 :
01B8 742 : All pertinent fields of the IRP are assumed to be valid.
01B8 743 :
01B8 744 : IPL = FIPL
01B8 745 :
01B8 746 : Outputs: R0-R4 Garbage
01B8 747 :
01B8 748 : --
01B8 749 ALT_START:
0210 8F BB 01B8 750 PUSHR #*M<R4,R9> : Save reg
0005 10 01BC 751 BSBB S$ : Process request
0210 8F BA 01BE 752 POPR #*M<R4,R9> : Restore regs
0005 05 01C2 753 RSB : Return to caller with garbage in R0
0000 01C3 754
0079D 30 01C3 755 S$: BSBW XLATE : Get CDB from IRPSW_CHAN
5D 50 E9 01C6 756 BLBC R0,ABORT_START : If LBC then error
0000 01C9 757 ASSUME CDB_V_RUN EQ 0
59 3A A9 E9 01C9 758 BLBC CDB_W_STS(R9),ABORT_START : If LBC then not in RUN state
0001 E0 01CD 759 BBS #IRPSW_FUNC -
57 2A A3 01CF 760 IRPSW_STS(R3),RCV_START : If BS then IOS_READ else IOS_WRITE
01D2 761
01D2 762 : Fall thru to XMT_START
01D2 763 :
```

```
01D2 765 .SBTTL XMT_START, Start Transmit Operation
01D2 766
01D2 767 :++
01D2 768 XMT_START - Start Transmit Operation
01D2 769
01D2 770
01D2 771 This routine is called to start a transmit operation. The tributary is
01D2 772 known to be up and running at this point. All status must be returned via
01D2 773 the IOSB.
01D2 774
01D2 775
01D2 776 Inputs: R3 = IRP address
01D2 777 R5 = UCB address
01D2 778 R9 = CDB address
01D2 779
01D2 780 IPL = FIPL
01D2 781
01D2 782 Outputs: R0 = status of transmit request
01D2 783
01D2 784 R5-R7 are preserved.
01D2 785
01D2 786 --
01D2 787 XMT_START:
51 32 A3 3C 01D2 788 MOVZWL IRPSL_BCNT(R3),R1 ; Pick up length
50 2C A3 D0 01D6 789 MOVL IRPSL_SVAPTE(R3),R0 ; Pick up head of buffer
52 60 D0 01DA 790 MOVL (R0),R2 ; Get beginning of user message
01DD 791 10$: ;
01DD 792 ; Add CI padding to keep beginning quadword aligned
01DD 793 ;
52 07 93 01DD 794 BITB #^X<07>,R2 ; Need padding ?
72 07 13 01E0 795 BEQL 20$ ; If EQL no
01 8E 01E2 796 MNEGB #1,-(R2) ; Pad
51 D6 01E5 797 INCL R1 ; Adjust byte count
F4 11 01E7 798 BRB 10$
01E9 799 20$: ;
01E9 800 ; Send it to SCS requesting that the datagram be returned when done.
01E9 801 ;
01E9 802 PUSHQ R2 ; Save user msg & IRP addresses
38 A9 51 B1 01EC 803 CMPW R1,CDB_W_BUFSIZ(R9) ; Msg size within bounds?
52 31 1A 01F0 804 BGTRU 60$ ; If GTRU then no
54 52 20 C2 01F2 805 SUBL #32,R2 ; Go to begining of PPD header
28 54 D1 01F9 806 SUBL3 R0,R2,R4 ; Get offset to top of buffer
08 A2 54 AE 01FC 807 CMPL R4,#CXBSC_HEADER-32 ; Is header big enough ?
0A A2 38 B0 01FE 808 BLSS 60$ ; If LSS then header too small
54 00B4 C5 D0 0202 809 MNEGW R4,8(R2) ; Neg. offset to top of buffer
0206 810 MOVW #DYN$C_CIDG,10(R2) ; Structure type
0208 811 MOVL UCBSL_PDT(R5),R4 ; Recover the PDT
0208 812 SEND_DG_BUF_REG #1- ; Control returns immediately
0208 813 CDB_L_CDT(R9),BUFFER=(SP)
08 50 E9 0218 814 BLBC R0,60$ ; If LBC, datagram not queued
0218 815 POPQ R2 ; Restore IRP address
1C B9 63 0E 021E 816 INSQUE (R3),@CDB_Q_XMT_IRP+4(R9) ; Queue IRP
05 0222 817 40$: RSB ; Return to await completion
0223 818 ;
0223 819 60$: POPQ R2 ; Restore IRP address
0226 820
0226 821 ABORT_START:
```

CNDRIVER
V04-000

- VAX/VMS DECnet-CI Class Driver N 11
XMT_START, Start Transmit Operation

16-SEP-1984 01:19:27
5-SEP-1984 00:11:06

VAX/VMS Macro V04-00
[DRIVER.SRC]CNDRIVER.MAR;1

Page 20
(17)

0686 31 0226 822 BRW ABORT_IRP_POST
0229 823

; Report SS\$_ABORT via IOSB

```
0229 825 .SBTTL RCV_START, Start Receive Operation
0229 826
0229 827 :++
0229 828 RCV_START - Start Receive Operation
0229 829 :
0229 830
0229 831 This routine is called to start a receive operation. The tributary is
0229 832 known to be up and running at this point. If IRPSL_SVAPTE is none zero
0229 833 then it is assumed to be system buffer to be added to the receive free list.
0229 834 All status must be returned via the IOSB.
0229 835
0229 836 Inputs: R3 = IRP address
0229 837 R5 = UCB address
0229 838 R9 = CDB address
0229 839
0229 840 IPL = FIPL
0229 841
0229 842 Outputs: R0 = return status of receive request
0229 843
0229 844 :--
0229 845 RCV_START:
0229 846 MOVL R9,R4 ; Transfer CDB pointer
0229 847 BSBW ADDRCLIST ; Add IRP buffer to free list
0229 848
0229 849 : Check to see if message is available
0229 850
0229 851 REMQUE @CDB_Q_RCV_MSG(R9),R2 ; Dequeue a received message
0229 852 BVS 100$ ; Br if none
0229 853 BRW FINISH_RCV_IO ; Complete the I/O request
0229 854 100$:
0229 855 : No message available. Queue IRP to await arrival of message.
0229 856
0229 857 INSQUE (R3),@CDB_Q_RCV_IRP+4(R9) ; Queue IRP to await message
0229 858 RSB ; Return
0229 859
```

54 59 D0 06D3 30

52 28 B9 0F 03 1D 0677 31

24 B9 63 0E 05


```
023D 861 .SBTTL SETMODE_FDT, Set mode I/O operation FDT routine
023D 862
023D 863
023D 864 ** SETMODE_FDT - Set mode I/O operation FDT routine
023D 865
023D 866
023D 867 Setup control parameters. Optionally startup/shutdown the device or one
023D 868 of the tributaries. The subfunction modifiers are as follows:
023D 869
023D 870 IOSM_CTRL - If set, request is for device. Else, for tributary.
023D 871 IOSM_STARTUP - Start device or establish tributary connection.
023D 872 IOSM_SHUTDOWN - Shutdown device or disconnect tributary.
023D 873
023D 874
023D 875 The QIO parameter for SETMODE is:
023D 876
023D 877 P2 = Optional address of buffer descriptor for extended characteristics
023D 878
023D 879
023D 880 Inputs: R3 = IRP address
023D 881 R4 = PCB address
023D 882 R5 = UCB address
023D 883 R6 = CCB address
023D 884 R7 = Function code
023D 885 AP = address of first QIO parameter
023D 886
023D 887 Outputs: R0 = status of setmode request
023D 888
023D 889 R3-R5 are preserved.
023D 890 R7-R9 = destroyed
023D 891
023D 892
023D 893 SETMODE_FDT: ; Setmode FDT processing
023D 894
023D 895 Copy the characteristics buffer, if any. No return on error.
023D 896 On return, there's a buffer attached to IRPSL SVAPTE containing a
023D 897 copy of the user buffer -- hence we cannot "abort" the QIO passed
023D 898 this point but must return all errors via the IOSB.
023D 899
023D 900 Upon return, the IPL has been raised to FIPL
023D 901
023D 902 BSBW GET_CHAR_WBUF ; Get P2 characteristics buffer
023D 903 ; - no return on error
023D 904 MOVZWL IRPSW_FUNC(R3),R7 ; Get full function code.
023D 905 BBC S^#IOSV_CTRL,R7,10$ ; Br if not controller request
023D 906 BRW SETMODE_CTRL ; Process controller request
023D 907 10$:
023D 908 ; Perform setmode request on a tributary
023D 909
023D 910 BSBW XLATE ; Get CDB address if any
023D 911 BBC S^#IOSV_SHUTDOWN,R7,40$ ; Branch if not trib shutdown
023D 912
023D 913 ; Shutdown tributary modifier specified -- always successful.
023D 914 ; Shutdown may complete ahead of other queued I/O for this tributary.
023D 915
023D 916 BLBC R0,FINISH_SUC ; If LBC then no CDB
023D 917 BSBW ZAP_CDB_R9 ; Do the dirty work
```

029D 30
57 20 A3 3C
03 57 09 E1
0113 31
0715 30
08 57 07 E1
6C 50 E9
0513 30

```
67 11 0258 918 BRB FINISH_SUC ; Always return "success"
      025A 919 40$:
      025A 920 : IOSM_STARTUP tributary modifier specified or no modifier.
      025A 921 : Validate the P2 buffer and its contents.
      025A 922 :
51 FE26 CF 9E 025A 923 MOVAB TRIB_PTM_TABLE,R1 ; Set address of verification table
      52 04 59 04 025F 924 CLRL R2 ; No status flags yet
      52 3A A9 E8 0261 925 BLBS R9,50$ ; If LBS then no CDB
      072E 3C 0264 926 MOVZWL CDB_W_STS(R9),R2 ; Get status flags
      67 50 30 0268 927 50$: BSBW VALIDATE_P2 ; Validate the P2 buffer
      E9 0268 928 BLBC R0,FINISH_REQ ; If LBC, report error via IOSB
      026E 929 :
      026E 930 : Check trib address. If this is a trib address change for this
      026E 931 : channel (in which case unconditionally give up the old CDB even if
      026E 932 : the QIO subsequently fails), or if there is no current CDB, then
      026E 933 : attempt to bind this channel to the CDB for the new trib address.
      026E 934 :
51 0474 8F 3C 026E 935 MOVZWL #NMASC_PCCI TRI,R1 ; Get trib address param i.d.
      0796 30 0273 936 BSBW UNPACK_P2_BUF ; From P2 buffer
      05 50 E8 0276 937 BLBS R0,60$ ; If LBS, trib was specified
      59 59 E8 0279 938 BLBS R9,FINISH_REQ ; If LBS, no CDB - return R0,R1
      16 11 027C 939 BRB 80$ ; No trib addr, use current CDB
      0E 59 E8 027E 940 60$: BLBS R9,70$ ; If LBS, no CDB
      3E A9 52 91 0281 941 CMPB R2,CDB_B_TRB_ADDR(R9) ; Address being changed ?
      0D 13 0285 942 BEQL 80$ ; If EQL no
      00E0 C542 B4 0287 943 CLRW UCBSW_VEC_CHAN(R5)[R2] ; Give-up previous CDB
      04DC 30 028C 944 BSBW ZAP_CDB_R9 ; Shut it down
      4A 10 028F 945 70$: BSBW NEW_TRIB ; Init/allocate CDB
      30 50 E9 0291 946 BLBC R0,FINISH_ERR ; If LBC, report error
      0294 947 80$:
      0294 948 : Tributary now exists change its characteristics and set them
      0294 949 : if trib is established.
      0294 950 :
      0294 951 :
29 57 06 E1 0294 951 BBC S^#IOSV STARTUP,R7,FINISH_SUC ; Br if not startup request
50 02C4 8F 3C 0298 952 MOVZWL #SS$ DEACTIVE,R0 ; Assume trib already active
51 3A A9 3C 029D 953 MOVZWL CDB_Q_STS(R9),R1 ; Get current status
      02A1 954 ASSUME CDB_C_IDLE EQ 0 ;
      51 3F A9 88 02A1 955 BISB CDB_B_STA(R9),R1 ; OR in the state
      51 30 A9 C8 02A5 956 BISL CDB_L_SETMODE(R9),R1 ; OR in pending SETMODE address
      19 12 02A9 957 BNEQ FINISH_ERR ; If NEQ then can't do startup
      30 A9 53 D0 02AB 958 MOVL R3,CDB_L_SETMODE(R9) ; Save IRP address
      027C 30 02AF 959 BSBW START_TRIB ; Startup the trib
      02B2 960 :
      02B2 961 : Fall thru to QIORET
```

```
.SBTTL Complete QIO request routines

02B2 963
02B2 964
02B2 965
02B2 966
02B2 967
02B2 968
02B2 969
02B2 970
02B2 971
02B2 972
02B2 973
02B2 974
02B2 975
02B2 976
00000000'GF 17 02B2 977 QIORET: JMP G^EXESQIORETURN ; Return success in R0 to user
50 2C D0 02B8 978 ABORT_REQ1: MOV S^#SS$ABORT,R0 ; Setup error status
00000000'GF 17 02B8 979 ABORT_REQ: MOV S^#SS$ABORT,R0 ; Setup error status
02B8 980 ABORT_REQ: JMP G^EXESABORTIO ; Exit QIO service with error
02C1 981
02C1 982
02C1 983
02C1 984
02C1 985
02C1 986
02C1 987
02C1 988
02C1 989
02C1 990
02C1 991
02C1 992
02C1 993
02C1 994
02C1 995
50 01 3C 02C1 996 FINISH_SUC: MOVZWL S^#SS$NORMAL,R0 ; Set success
51 D4 02C4 997 FINISH_ERR: CLRL R1 ; Clear second IOSB longword
0A 20 A3 E0 02C6 998 BBS S^#IOSV_CTRL,- ; Skip for controllers
07 50 E9 02CB 1000 IRPSW_FUNC(R$),FINISH_REQ
51 00002800 BF D0 02CB 1001 ASSUME CDB V-RUN EQ 0
02D5 1002 BLBC R0,FINISH_REQ ; If LBC then circuit not up
02D5 1003 MOVL #XMSM_STS_ACTIVE!- ; Indicate circuit up
02D5 1004 XMSM_STS_RUNNING,R1
00000000'GF 17 02D5 1005 FINISH_REQ:
02DB 1006 JMP G^EXESFINISHIO ; Complete the I/O
02DB 1007
```

The following routines all exit the \$QIO system service with status in R0.
If an error is not being returned, further status will be eventually passed
via the IOSB when the IRP undergoes post processing.

Inputs: R3 IRP address
R5 UCB address
IPL may be FIPL or IPL\$ASTDEL

The following routines exit the \$QIO system service with SS\$ NORMAL and
send the IRP back to IOPOST to return final status via the IOSB.

Inputs: R3 IRP address
R5 UCB address
IPL may be either FIPL or IPL\$ASTDEL

```
.SBTTL NEW_TRIB - Allocate and init new CDB

NEW_TRIB - allocate and init new CDB

A CDB is allocated and initialized and stored in the UCB CDB vector.
The address is also stored in IRPSL_CDB.

Inputs:      R2 Trib address
             R3 IRP address
             R5 UCB address

Outputs:     R9 CDB address

R0-R2 are clobbered. All other registers are preserved.

NEW_TRIB:
    MOVB R2,IRPSB_INDEX(R3)          ; Set the trib number
    MOVZWL #SS$DEVXLRALLOC,R0        ; Assume error
    TSTW UCB$Q_VEC_CHAN(R5)[R2]      ; Claimed by another channel?
    BNEQ 40$                          ; If NEQ yes, report error
    MOVL UCB$L_VEC_CDB(R5)[R2],R9     ; Get associated CDB
    BNEQ 30$                          ; If NEQ, CDB exists - claim it
    BSBB NEW_CDB                     ; Create a new CDB
    BLBC R0,40$                      ; If LBC then error
    MOVZBL IRPSB_INDEX(R3),R2        ; Restore trib address
    MOVL R9,UCB$L_VEC_CDB(R5)[R2]    ; Store CDB address in UCB
    MOVW IRPSW_CHAN(R3),-            ; Save channel index in UCB
    UCB$W_VEC_CHAN(R5)[R2]
    30$: MOVW
    40$: RSB                          ; Done

NEW_CDB:
    MOVZWL #CDB_C_LENGTH,R1          ; Create new CDB
    PUSHL R3                          ; Get size of CDB
    JSB G^EXESALONONPAGED            ; Save reg
    POPL R3                           ; Allocate the CDB
    BLBC R0,100$                     ; Restore reg
    ; Br if error
    ; Initialize CDB
    MOVL R2,R9                        ; Copy CDB address
    MOVL R2,IRPSL_CDB(R3)             ; Save it in IRP
    MOVAB CDB_W_SIZE(R2),R2          ; Setup ptr to init CDB
    ASSUME CDB_B_TYPE EQ 2+CDB_W_SIZE
    ASSUME CDB_B_FIPL EQ 1+CDB_B_TYPE
    ASSUME CDB_L_FPC EQ 1+CDB_B_FIPL
    ASSUME CDB_L_FR3 EQ 4+CDB_L_FPC
    ASSUME CDB_L_FR4 EQ 4+CDB_L_FR3
    MOVW R1,(R2)+                     ; CDB_W_SIZE
    MOVW #<CDB_C_FIPL>+DYN$C_NET,(R2)+ ; CDB_B_TYPE and CDB_B_FIPL
    ADDL #3+4,R2                      ; Advance passed CDB_L_FR4
```



```

0333 1066
0333 1067
0333 1068
0333 1069
0333 1070
51 03 D0 0333 1071
62 62 DE 0333 1072 20$:
82 82 DE 0336 1073
F7 51 F5 0339 1074
033C 1075
033F 1076
033F 1077
033F 1078
033F 1079
033F 1080
033F 1081
82 42 A5 7C 0341 1082
3C 0345 1083
0345 1084
0345 1085
0345 1086
0345 1087
0345 1088
0345 1089
62 009F C5 90 0345 1090
82 82 90 034A 1091
82 40 A3 9B 034D 1092
0351 1093
0351 1094
0351 1095
0351 1096
0351 1097
0351 1098
82 7C 0351 1099
82 7C 0353 1100
0355 1101
0355 1102
0355 1103
0355 1104
82 55 D0 0355 1105
82 82 D4 0358 1106
50 01 D0 035A 1107
05 035D 1108 100$:
035E 1109
035E 1110

ASSUME CDB_Q_XMT_IRP EQ 4+CDB_L_FR4
ASSUME CDB_Q_RCV_IRP EQ 8+CDB_Q_XMT_IRP
ASSUME CDB_Q_RCV_MSG EQ 8+CDB_Q_RCV_IRP

MOVL #3,R1
MOVAL (R2),(R2)
MOVAL (R2)+(R2)+
SOBGR R1,20$

ASSUME CDB_L_SETMODE EQ 8+CDB_Q_RCV_MSG
ASSUME CDB_L_ABSTIME EQ 4+CDB_L_SETMODE
ASSUME CDB_W_BUFSIZ EQ 4+CDB_L_ABSTIME
ASSUME CDB_W_STS EQ 2+CDB_W_BUFSIZ

CLRQ (R2)+
MOVZWL UCB$W_DEVBUFSIZ(R5),(R2)+

ASSUME CDB_B_RCV_CNT EQ 2+CDB_W_STS
ASSUME CDB_B_RCV_FQ EQ 1+CDB_B_RCV_CNT
ASSUME CDB_B_TRB_ADDR EQ 1+CDB_B_RCV_FQ
ASSUME CDB_B_STA EQ 1+CDB_B_TRB_ADDR
ASSUME CDB_C_IDLE EQ 0

MOVB UCB$B_RCV_CNT(R5),(R2)
MOVB (R2)+(R2)+
MOVZBL IRP$B_INDEX(R3),(R2)+

ASSUME CDB_L_BRC EQ 1+CDB_B_STA
ASSUME CDB_L_BSN EQ 4+CDB_L_BRC
ASSUME CDB_L_DBR EQ 4+CDB_L_BSN
ASSUME CDB_L_DBS EQ 4+CDB_L_DBR

CLRQ (R2)+
CLRQ (R2)+

ASSUME CDB_L_UCB EQ 4+CDB_L_DBS
ASSUME CDB_L_CDT EQ 4+CDB_L_UCB

MOVL R5,(R2)+
CLRL (R2)+
MOVL #1,R0
RSB

: Set number of queue heads
: Init forward link pointer
: Init backward link pointer
: Loop if more queues
: Init CDB L SETMODE,ABSTIME
: CDB_W_BUFSIZ and CDB_W_STS
: CDB_B_RCV_CNT (default)
: CDB_B_RCV_FQ (default)
: CDB_B_TRB_ADDR, CDB_B_STA
: CDB_L_BRC and CDB_L_BSN
: CDB_L_DBR and CDB_L_DBS
: CDB_L_UCB
: CDB_L_CDT
: Indicate success
: Done
```

```
035E 1112 .SBTTL SETMODE_CTRL, Perform setmode FDT operation on controller
035E 1113
035E 1114
035E 1115 SETMODE_CTRL - Perform setmode FDT operation on controller
035E 1116
035E 1117
035E 1118 This routine performs the SETMODE FDT setup for the controller.
035E 1119
035E 1120 Inputs: R3 = IRP address
035E 1121 R4 = PCB address
035E 1122 R5 = UCB address
035E 1123 R7 = IRP function word
035E 1124
035E 1125 Outputs: R0 = status of setmode request
035E 1126
035E 1127 R3-R5 are preserved.
035E 1128
035E 1129
035E 1130 SETMODE_CTRL:
035E 1131 BBC S^#IOSV_SHUTDOWN,R7,10$ ; Perform setmode on controller
0362 1132 ; Br if not shutdown request
0362 1133
0362 1134 Shutdown modifier specified
0362 1135
0362 1136 BSBW CAN_DEV ; Shutdown the device
0365 1136 BRB 50$ ; Finish the QIO with success
0367 1137 10$:
0367 1138 Startup line modifier specified or no modifier
0367 1139
0367 1140 BBS #UCBSV_CN_INITED,- ; Br if controller up already
0369 1141 UCB$W_DEVSTS(R5),40$
036C 1142
036C 1143 Validate P2
036C 1144
036C 1145 MOVAB LINE_PRM_TABLE,R1 ; Address of verif table
0371 1146 MOVL R5,R9 ; Address of current param's
0374 1147 MOVZWL UCB$W_DEVSTS(R5),R2 ; Status flags
0378 1148 BSBW VALIDATE_P2 ; Validate P2 buffer
037B 1149 BLBC R0,70$ ; If LBC, return R0,R1 in IOSB
037E 1150
037E 1151 Setup Maximum receive buffers
037E 1152
037E 1153 MOVZWL #NMASC_PCLI_BFN,R1 ; Set to find MAX RCV
0383 1154 BSBW UNPACK_P2_BOF ; In P2 buffer
0386 1155 BLBC R0,30$ ; Br if not found
0389 1156 MOVB R2,UCB$B_RCV_CNT(R5) ; Initialize number of RCV
038E 1157 30$:
038E 1158
038E 1159 Setup Blocksize
038E 1160
038E 1161 MOVZWL #NMASC_PCLI_BUS,R1 ; Get buffer size
0393 1161 BSBW UNPACK_P2_BOF ; From P2 buffer
0396 1162 BLBC R0,40$ ; Br if not found
0399 1163 MOVW R2,UCB$W_DEVBUSIZ(R5) ; and in UCB
039D 1164 40$:
039D 1165
039D 1166 Device initialized - then do a LISTEN if IOSV_STARTUP
039D 1167
039D 1167 BBC S^#IOSV_STARTUP,R7,50$ ; Finish up if not starting
03A1 1168 BSBB LISTEN ; Do a LISTEN
```

```
06 50 E9 03A3 1169 BLBC R0,100$ ; If LBC then failed
FF18 31 03A6 1170 50$: BRW FINISH_SUC ; Finish - SS$ NORMAL for IOSB
FF29 31 03A9 1171 70$: BRW FINISH_REQ ; Finish - R0,R1 for IOSB
FF0C 31 03AC 1172 100$: BRW ABORT_REQ ; Abort the I/O request
03AF 1173
03AF 1174 LISTEN: ;
03AF 1175 ; Do all the wonderful SCS magic needed to start up. The buffer
03AF 1176 ; created on the stack is pointed to by R7 in case the CONFIG_SYS
03AF 1177 ; macro is updated someday to modify SP as it pushes arguments.
03AF 1178 ;
03AF 1179 ; NOTE: The following code assumes that we have only 1 CI port on
03AF 1180 ; the current system!!!!
03AF 1181
03AF 1182
00000070 03AF 1183 SBO_LNG = SBO$C_LENGTH + 32 ; SBO length plus random amount
03AF 1184 ; of padding merely for merely
03AF 1185 ; hysterical purposes.
59 53 D0 03AF 1186 MOVL R3,R9 ; Save R3
SE 00000070 8F C2 03B2 1187 SUBL #SBO_LNG,SP ; Create buffer on stack
57 5E D0 03B9 1188 MOVL SP,R7 ; Preserve value of buffer
08 50 E9 03BC 1189 CONFIG_SYS G^SCS$GB_SYSTEMID,(R7) ; Get our system block
3C A7 90 03CC 1191 BLBC R0,200$ ; If LBC, not ready yet
009E C5 07 18 03CD 1192 MOV B SBO$B_RSTATION1(R7),- ; Get our port number
UCB$B_CN_PORT(R5)
50 0084 8F 3C 03D5 1194 BGEQ 210$ ; If LSS then not ready yet
08 36 11 03D7 1195 MOVZWL #SS$DEVOffline,R0 ; Device offline error (no PA)
56 14 A1 D0 03DC 1196 BRB 220$ ; Exit
56 2C A6 D0 03DE 1197 210$: MOVL SBL_PBCONNX(R1),R6 ; Get path block
0084 C5 56 D0 03E2 1198 MOVL PBL_PDT(R6),R6 ; Pick up PDT
03E6 1199 MOVL R6,UCB$P_PDT(R5) ; Save in UCB
03EB 1200 LISTEN - ; Setup a LISTEN
03EB 1201 MSGADR = W^LIS_FORK,-
03EB 1202 ERRADR = W^LIS_ERR,-
03EB 1203 LPRNAM = PROC_NAM,-
03EB 1204 PRINFO = PROC_NAM
0D 50 E9 0404 1205 BLBC R0, 220$ ; If LBC then error
0090 C5 53 D0 0407 1206 MOVL R3,UCB$P_LIS_CDT(R5) ; Save listen CDT
5C A3 55 D0 040C 1207 MOVL R5,CDT$P_AUXSTRUC(R3) ; Set addr of UCB into CDT
68 A5 01 A8 0410 1208 BLSW #UCB$M_CN_INITED,UCB$W_DEVSTS(R5) ; Indicate device initd
0414 1209
SE 00000070 8F C0 0414 1210 220$: ADDL #SBO_LNG,SP ; Restore stack
53 59 D0 041B 1211 MOVL R9,R3 ; Restore IRP addr
05 041E 1212 RSB
041F 1213
041F 1214 LIS_ERR: ; Error on LISTEN CDT
041F 1215 DISCONNECT ; Put it back to listen
05 0425 1216 RSB ; Leave
0426 1217
```

```

0426 1219 .SBTTL SENSEMODE_FDT, Sense Mode I/O operation FDT routine
0426 1220
0426 1221 ++
0426 1222 : SENSEMODE_FDT - Sense Mode FDT routine
0426 1223
0426 1224 This routine returns information to the caller about the configuration
0426 1225 and status of the CI device. Depending on the function modifier,
0426 1226 either the device characteristics or error counters contents are returned.
0426 1227
0426 1228 The QIO parameters for SENSEMODE are:
0426 1229
0426 1230 P2 = optional address of buffer descriptor for extended characteristics
0426 1231
0426 1232
0426 1233 Inputs: R3 = IRP address
0426 1234 R4 = PCB address
0426 1235 R5 = UCB address
0426 1236 R6 = CCB address
0426 1237 R7 = Function code
0426 1238 AP = Address of first function-dependent QIO parameter
0426 1239
0426 1240 Outputs: R0 = status return of sensemode request
0426 1241
0426 1242 R3-R5 are preserved.
0426 1243
0426 1244 --
0426 1245 SENSE_TABLE:
03A2 0426 1246 .WORD SENSE_TABLE - TRIB_PRM_TABLE ; Setup list of offset to
034A 0428 1247 .WORD SENSE_TABLE - TRIB_CNT_TABLE ; parameter tables with using
037C 042A 1248 .WORD SENSE_TABLE - LINE_PRM_TABLE ; the following 2 bit index:
0338 042C 1249 .WORD SENSE_TABLE - LINE_CNT_TABLE ;
042E 1250 ; bit 0 set => counters
042E 1251 ; bit 1 set => non-trib
042E 1252 SENSEMODE_FDT: ; Sensemode FDT I/O processing
00000080 042E 1253 SENSE_C_BUF = 128
042E 1254
042E 1255 ASSUME TRIB_PRM_NUM*6 LE SENSE_C_BUF ; Make sure buffer can hold all
042E 1256 ASSUME LINE_PRM_NUM*6 LE SENSE_C_BUF ; info for all cases
042E 1257 ASSUME TRIB_CNT_NUM*6 LE SENSE_C_BUF
042E 1258 ASSUME LINE_CNT_NUM*6 LE SENSE_C_BUF
042E 1259
042E 1260 ; Check user buffer. Get system buffer. Setup IRP
042E 1261
042E 1262 MOVW #SENSE_C_BUF,IRPSL_IOST1+2(R3) ; Setup buff size needed
3A A3 0080 8F B0 0434 1263 BSBW GET_CHAR_RBUF ; Setup "read" buff for IOPOST
00A2 30 0437 1264 ; - no return on error
0437 1265 MOVZWL IRPSW_FUNC(R3),R7 ; Get full function code.
04 57 20 A3 3C 0437 1265 MOVZWL IRPSW_FUNC(R3),R7 ; Get full function code.
04 A2 3C A3 D0 043B 1266 MOVL IRPSL_IOST2(R3),4(R2) ; Store user buffer virt addr
0440 1267 ; in standard place in buffer
52 62 D0 0440 1268 MOVL (R2),R2 ; Get pointer to data area
0443 1269
0443 1270 ; Locate parameter/counter table
0443 1271
0443 1272 MOVL #3,R6 ; Init SENSE TABLE index
56 03 D0 0443 1272 MOVL #3,R6 ; Init SENSE TABLE index
58 02 D0 0446 1273 MOVL #COUNT_C_ENTRY-2,R8 ; Bias COUNTER table entry size
09 57 08 E0 0449 1274 BBS #IOSV_RD_COUNT,R7,108 ; If BS, "read counter" request
56 97 044D 1275 DECB R6 ; Erase "read counter" bit

```



```
00 58 0A D0 044F 1276      MOVL    #PARAM_C_ENTRY-2,R8      ; Bias PARAM table entry size
    57 08 E5 0452 1277      BBCC    #IOSV_RD_COUNT,R7,10$      ; Clear out garbage modifier
    59 55 D0 0456 1278 10$: MOVL    R5,R9      ; If IOSV_CTRL, use UCB source
09 57 09 E0 0459 1279      BBS     #IOSV_CTRL,R7,20$      ; If BS, not for a tributary
    56 02 8A 045D 1280      BICB    #2,R6      ; Erase "non-tributary" flag
    0500 30 0460 1281      BSBW    XLATE      ; Locate CDB, use CDB source
    6C 50 E9 0463 1282      BLBC    R0,100$      ; If LBC then CDB not found
50 BC AF 46 32 0466 1283 20$: CVTWL  SENSE_TABLE[R6],R0      ; Get offset to parameter table
    56 B8 AF 9E 046B 1284      MOVAB  SENSE_TABLE,R6      ; Get address of base
    56 50 C2 046F 1285      SUBL    R0,R6      ; Calculate table address
    0472 1286 30$:      ;
    0473 1287      ; Fill buffer with requested information
    0474 1288      ;
    51 86 B0 0472 1289      MOVW    (R6)+,R1      ; Get parameter i.d.
    2E 13 0473 1290      BEQL    60$      ; If EQL, at end of table
    05 57 08 E0 0477 1291      BBS     #IOSV_RD_COUNT,R7,40$      ; If BS then counter i.d.
54 51 F000 8F AA 047B 1292      BICW    #*C<PRM M TYPE>,R1      ; Else param i.d., clear junk
    66 0A 00 EF 0480 1293 40$: EXTZV  #OFF_V_VALUE,#OFF_S_VALUE,(R6),R4      ; Get source offset
    66 06 0A EF 0483 1294      EXTZV  #OFF_V_WIDTH,#OFF_S_WIDTH,(R6),R0      ; Get source width
    14 13 048A 1295      BEQL    50$      ; If EQL, ignore this param
    54 59 C0 048C 1296      ADDL    R9,R4      ; Calculate source address
    82 82 51 B0 048F 1297      MOVW    R1,(R2)+      ; Enter parameter i.d.
    64 50 00 EF 0492 1298      EXTZV  #0,R0,(R4),(R2)+      ; Enter parameter value
    05 57 0A E1 0497 1299      BBC     #IOSV_CLR_COUNT,R7,50$      ; If BC, don't clear source
    64 50 00 F0 049B 1300      INSV    #0,#0,R0,(R4)      ; Clear counter
    56 58 C0 04A0 1301 50$: ADDL    R8,R6      ; Advance to next entry
    CD 11 04A3 1302      BRB     30$      ; Loop
    04A5 1303 60$:      ;
    04A5 1304      ; Setup status and transfer size
    04A5 1305      ;
    52 2C B3 C2 04A5 1306      SUBL    @IRPSL_SWAPTE(R3),R2      ; Calculate bytes moved
00000080 8F 52 D1 04A9 1307      CMPL    R2,#SENSE_C_BUF      ; Was our buffer large enough ?
    23 1A 04B0 1308      BGTRU    200$      ; If GTRU no
    38 A3 01 B0 04B2 1309      MOVW    #$$$_NORMAL,IRPSL_IOST1(R3)      ; Assume success
    52 32 A3 B1 04B6 1310      CMPW    IRPSW_BCNT(R3),R2      ; User buffer big enough ?
    0A 1E 04BA 1311      BGEQU    80$      ; If GEQU then yes
    38 A3 0601 8F B0 04BC 1312      MOVW    #$$$_BUFFEROVF,IRPSL_IOST1(R3)      ; Show warning
    52 32 A3 B0 04C2 1313      MOVW    IRPSW_BCNT(R3),R2      ; Shrink xfer size
    3A A3 52 B0 04C6 1314 80$: MOVW    R2,IRPSL_IOST1+2(R3)      ; Move xfer size to IOSB image
    32 A3 52 B0 04CA 1315      MOVW    R2,IRPSW_BCNT(R3)      ; Setup xfer size for IOPOST
    50 38 A3 D0 04CE 1316      MOVL    IRPSL_IOST1(R3),R0      ; Set length/status
    FDEF 31 04D2 1317 100$: BRW     FINISH_ERR      ; Leave setting R0 in IOSB
    04D5 1318      ;
    04D5 1319 200$: BUG_CHECK INCONSTATE,FATAL      ; We've corrupted pool
```

```
04D9 1321 .SBTTL GET_CHAR_RBUF, Get P2 characteristics buffer for read
04D9 1322 .SBTTL GET_CHAR_WBUF, Get P2 characteristics buffer for write
04D9 1323
04D9 1324 :++
04D9 1325 GET_CHAR_RBUF - Get P2 characteristics buffer for read
04D9 1326 GET_CHAR_WBUF - Get P2 characteristics buffer for write
04D9 1327
04D9 1328 This routine saves the address of P2 buffer for later use by the driver.
04D9 1329 The P2 buffer address is saved in IRPSL_IOST2 of the IRP, and the size
04D9 1330 in IRPSL_BCNT.
04D9 1331
04D9 1332 Inputs: R3 = IRP address
04D9 1333 R4 = PCB address
04D9 1334 R5 = UCB address
04D9 1335
04D9 1336 Outputs: R0 = Garbage
04D9 1337 R1 = User buffer size
04D9 1338 R3-R5 are preserved.
04D9 1339
04D9 1340 :--
04D9 1341 GET_CHAR_RBUF:
04D9 1342 BISB #IRPSM_FUNC,IRPSW_STS(R3) : Get P2 char buffer for 'read'
04DD 1343 GET_CHAR_WBUF: : Mark IRP for 'read'
04DD 1344 CLRQ R0 : Get P2 char buffer for 'write'
04DF 1345 MOVL P2(AP),R2 : Setup null user buffer
04E3 1346 BEQL 10$ : Get address of P2 desc
04E5 1347 EXTZV #0,#2,IRPSB_RMOD(R3),R0 : If EQL, no P2 was specified
04EB 1348 IFNORD #8,(R2),50$,MODE=R0 : Get access mode
04F1 1349 MOVZWL (R2),R1 : Br if no read access
04F4 1350 MOVL 4(R2),R0 : Get buffer length in bytes
04F8 1351 10$: MOVL R0,IRPSL_IOST2(R3) : Get buffer address
04FC 1352 TSTW R1 : Save it for later
04FE 1353 BEQL 30$ : Null user buffer?
0500 1354 PUSHAB B*30$ : If EQL yes, don't probe
0503 1355 BBC #IRPSV_FUNC,IRPSW_STS(R3),20$ : Setup return address
0508 1356 JMP G*EXES$READCHK : If BC then 'write'
050E 1357 20$: JMP G*EXES$WRITECHK : Check user buffer, setup IRP
0514 1358 : Check user buffer, setup IRP
0514 1359 30$: BBC #IRPSV_FUNC,IRPSW_STS(R3),40$ : - no return on error
0519 1360 MOVZWL IRPSL_IOST1+2(R3),R1 : If BC then 'write'
051D 1361 40$: BSBW GET_BOF : Get required buffer size
0520 1362 SETIPL UCBSB_FIPL(R5) : Get buffer
0524 1363 BLBS R0,60$ : Raise IPL
0527 1364 50$: MOVZWL #SS$_ACCVIO,R0 : Okay if LBS
052A 1365 BRW ABORT_REQ : Set error status
052D 1366 60$: RSB : Abort the I/O request
052E 1367
```

2A A3 02 88 04D9 1342
52 04 AC 7C 04DD 1343
13 13 04DF 1345
50 0B A3 02 00 EF 04E3 1346
51 62 3C 04E5 1347
50 04 A2 DO 04EB 1348
3C A3 50 DO 04F1 1349
51 B5 04F4 1350
14 13 04F8 1351
14 AF 9F 04FC 1352
06 2A A3 01 E1 04FE 1353
00000000 GF 17 0500 1354
00000000 GF 17 0503 1355
04 2A A3 01 E1 0508 1356
51 3A A3 3C 050E 1357
FCOD 30 0514 1358
06 50 E8 0519 1359
50 OC 3C 051D 1360
FD8E 31 0520 1361
05 0524 1362
0527 1363
052A 1364
052D 1365
052E 1366

```
052E 1369 .SBTTL START_TRIB, Start tributary routine
052E 1370
052E 1371
052E 1372
052E 1373
052E 1374
052E 1375
052E 1376
052E 1377
052E 1378
052E 1379
052E 1380
052E 1381
052E 1382
052E 1383
052E 1384
052E 1385
052E 1386
052E 1387
052E 1388
052E 1389
052E 1390
052E 1391
052E 1392
052E 1393
052E 1394
052E 1395
052E 1396
052E 1397
052E 1398
052E 1399
052E 1400
052E 1401
052E 1402
052E 1403
052E 1404
052E 1405
052E 1406
052E 1407
052E 1408
052E 1409
052E 1410
052E 1411
052E 1412
052E 1413
052E 1414
052E 1415
052E 1416
052E 1417
052E 1418
052E 1419
052E 1420
052E 1421
052E 1422
052E 1423
052E 1424
052E 1425

51 0479 8F 3C 052E 1392 MOVZWL #NMASC_PCCI MRB,R1 ; Get param i.d.
    04D6 30 0533 1393 BSBW UNPACK_P2_BUF ; Get param value
    05 50 EB 0536 1394 BLBS R0,20$ ; If LBS then param was there
52 009F C5 9A 0539 1395 MOVZBL UCB$B_RCV_CNT(R5),R2 ; Else, get default
    09 52 91 053E 1396 20$: CMPB R2,#RBFMIN ; Are there enough buffers to
    03 1A 0541 1397 BGTRU 30$ ; reduce datagram loss?
    52 09 D0 0543 1398 ; If not, do user a favor
    52 1F D1 0546 1399 30$: CMPL #RBFMAX,R2 ; Too many?
    03 1A 0549 1400 BGTRU 40$ ; If GTRU then no
    52 1F D0 054B 1401 MOVL #RBFMAX,R2 ; Use safer minimum
    3C A9 52 90 054E 1402 40$: MOVB R2,CDB_B_RCV_CNT(R9) ; Setup receive pool accounting
    3D A9 52 90 0552 1403 MOVB R2,CDB_B_RCV_FQ(R9) ; List starts out full
    0556 1404
    0556 1405 ; Init CDB state.
    0556 1406
48 A3 30414150 8F D0 0556 1407 MOVL #A/PAA0/,IRP$B_INDEX+8(R3) ; Set to connect over local
    3F A9 03 90 055E 1408 ; port PAA0
    009E C5 91 055E 1409 MOVB #CDB_C_LSTN,CDB_B_STA(R9) ; Assume "listen" state
    40 A3 91 0562 1410 CMPB UCB$B_CN_PORT(R5),- ; Compare our address to
    22 1F 0566 1411 IRP$B_INDEX(R3) ; remote's address
    04 1A 0568 1412 BLSSU 100$ ; If LSSU, stay in "listen"
    52 D4 056A 1413 BGTRU 50$ ; If GTRU, initiate connect
    04 11 056C 1414 CLRL R2 ; Else we're talking to
    3F A9 02 90 056E 1415 ; ourselves -- zero rcv buffers
    14 3A A9 01 E2 0570 1416 50$: MOVB #CDB_C_CONN,CDB_B_STA(R9) ; CONNECT from "LSTN" state
    55 DD 0579 1417 60$: BBSS #CDB_V_CONN,CDB_W_STS(R9),200$ ; Else, go to "connect" state
    55 59 D0 057B 1418 ; Indicate waiting return from
    11 10 057E 1419 CONNECT
    55 55 DD 0579 1420 PUSHL R5 ; Save UCB address
    55 59 D0 057B 1421 MOVL R9,R5 ; Use CDB for CONNECT context
    55 11 10 057E 1422 BSBB CONN ; Post connect request to SCS
    04 3A A9 01 E1 0580 1423 POPL R5 ; Restore UCB address
    54 A9 53 D0 0583 1424 BBC #CDB_V_CONN,CDB_W_STS(R9),100$ ; If BC, completed synchronously
    54 A9 53 D0 0588 1425 MOVL R3,CDB_L_CDI(R9) ; Set pointer to open CDI
```



```
05 058C 1426 100$: RSB ; Done
058D 1427
058D 1428 200$: BUG_CHECK INCONSTATE,FATAL ; Bug if already set
0591 1429
50 3C A9 9A 0591 1430 CONN: MOVZBL CDB_B_RCV_CNT(R9),R0 ; Pick up rcv buffer count
0595 1431 CONNECT - ; Request a CONNECT
0595 1432 MSGADR = W*MSG_FORK,- ; Message address
0595 1433 DGADR = W*DG_FORK,- ; Psuedo interrupt routine
0595 1434 ERRADR = W*CONN_ERR,- ; Connect errors
0595 1435 RSYSD = 0,- ; No remote system specified
0595 1436 RSTADR = IRPSB_INDEX(R3),- ; Virtual circ to connect over
0595 1437 RPRNAM = PROC_NAM,- ; To whom we will speak
0595 1438 LPRNAM = PROC_NAM,- ; Our name
0595 1439 INITCR = #1,- ; Allow for messages
0595 1440 INITDG = R2,- ; Number of receive buffers
0595 1441 CONDAT = CONN_DATA,- ; Connect data
0595 1442 AUXSTR = (R5) ; Auxiliary structure
05CB 1443
05CB 1444 ;
05CB 1445 ; Control returns to caller's caller - the JMP G*EXESQIORETURN.
05CB 1446 ; When the connection completes, the following is called as a fork
05CB 1447 ; process NOT necessarily in the context of process.
05CB 1448
05CB 1449 R0 = Status code
05CB 1450 R1 = Reject reason if status = reject
05CB 1451 R2 --> ACCEPT_REQ msg if status = success
05CB 1452 R3 --> Connection CDT
05CB 1453 R4 --> PDT
05CB 1454 R5 --> CDB
05CB 1455
05CB 1456 CLRBIT #CDB_V_CONN,CDB_W_STS(R5) ; No longer awaiting CONN return
OE 50 E9 05D0 1457 BLBC R0,20$ ; If LBS then error
00F9 30 05D3 1458 BSBW CHECK_REMOTE ; Check remote's connect info
03 12 05D6 1459 BNEQ 10$ ; If NEQ, can't talk to remote
00B8 31 05D8 1459 BRW CONN_FIN ; Else okay, complete setup
54 A5 D4 05DB 1460 10$: DISCONNECT ; Break the connection
05E1 1461 20$: CLRL CDB_L_CDT(R5) ; Forget about CDT, if any
05E4 1462
05E4 1463 CONN_ABO:
05E4 1464
05E4 1465
05E4 1466
05E4 1467
05E4 1468 CONNECT or ACCEPT failed.
05E4 1469
05E4 1470 If we were to return an error immediately every time, the higher
05E4 1471 level user (NETACP) would consume too much time trying to restart
05E4 1472 the circuit. This is because the CI, unlike other devices, will
05E4 1473 return immediately if the partner is not ready on a CONNECT
05E4 1474 attempt. For all other devices, the connect remains pending
05E4 1475 indefinitely.
05E4 1476
05E4 1477 Therefore, in order to save CPU cycles, simply return and allow the
05E4 1478 IOSM_STARTUP $QIO to hang indefinitely. This forces NETACP to
05E4 1479 initiate the subsequent cleanup via a $CANCEL, $DASSGN, or
05E4 1480 IOSM_SHUTDOWN. This should be fixed someday to have CNDRIVER retry
05E4 1481 every 5 seconds or so without reporting an error.
05E4 1482
*** NOTE: This logic here has been retained in case it needs
to be reactivated someday. However, it has been
```



```

05E4 1483
05E4 1484
05E4 1485
05E4 1486
05E4 1487
05E4 1488
05E4 1489
05E4 1490
05E4 1491
05E7 1492
05E9 1493
05ED 1494
05F2 1495
05F3 1496
05F9 1497
05F9 1498
05FA 1499
05FB 1500
05FE 1501
0601 1502
0602 1503
0602 1504

3A A5 B5
18 12
50 50 A5 D0
64 A0 05 E0
5B A5 96
AS 03 93

01
01
54 55 D0
016D 30
05 0601
0602
0602

TSTW
BNEQ
MOVL
BBS
INCB
BITB
BEQL
NOP
NOP
MOVL
BSBW
RSB

CDB_W_STS(R5)
100$
CDB_L UCB(R5),R0
#UCB$V POWER,UCB$W_STS(R0),50$
CDB_B_RSTCNT(R5)
#3,CDB_B_RSTCNT(R5)
100$

R5,R4
ZAP_CDB

: All quiet yet ?
: If NEQ, just wait
: Get UCB address
: If BS, powerfial recovery
: Another restart attempt
: Is this the 4th phase ?
: If EQL yes, wait.

: Copy CDB address
: Report the error immediately
: Wait the gio until contacted
: by user via $CANCEL, etc

```

found that not returning an error immediately can cause some confusion since it can delay a circuit initialization for 3 minutes or so in some cases. In addition, the time spent by NETACP to continually reinitialize the circuit has been found to be small enough that it presents no real problem.

```
0602 1506 .SBTTL LIS_FORK, Listen action routine
0602 1507
0602 1508
0602 1509 **
0602 1510 LIS_FORK - Listen action routine
0602 1511
0602 1512 This routine is entered as a fork process activated by the PADRIVER
0602 1513 when some other process has sent a CONNECT to us. We can then decide
0602 1514 to ACCEPT or REJECT the connection.
0602 1515
0602 1516 Inputs: R2 --> CONNECT_REQ message
0602 1517 R3 --> Listening CDT
0602 1518 R4 --> PDT
0602 1519
0602 1520 IPL = FIPL
0602 1521
0602 1522
0602 1523 LIS_FORK:
0602 1524 MOVL CDT$AUXSTRUC(R3),R5 ; Pick up UCB from Listen CDT
0602 1525 MOVZBL CDT$B_RSTATION(R3),R1 ; Get other guy's port
0602 1526 MOVL UCBSL_VEC_CDB(R5)[R1],R5 ; Pick up the CDB
0602 1527 BEQL REJECT ; We don't have one, reject
0602 1528 CMPB CDB_B_STA(R5),#CDB_C_LSTN ; Are we listening on this trib
0602 1529 BNEQ REJECT ; If NEQ no, reject connection
0602 1530 BSBW CHECK_REMOTE ; Process connect data
0602 1531 BEQL ACCEPT ; If NEQ then from DECnet SYSAP
0602 1532 REJECT:
0602 1533 ; REJECT the connection.
0602 1534
0602 1535 CLRL R5 ; Forget about CDB
0602 1536 MOVL #SS$NORMAL,R0 ; Reject reason
0602 1537 REJECT ; Yes, reject him - return to
0602 1538 ; caller's caller.
0602 1539 RSB ; Return to SCS (nop)
0602 1540
0602 1541 ACCEPT:
0602 1542 ; ACCEPT the connection.
0602 1543
0602 1544 MOVZBL CDB_B_RCV_CNT(R5),R0 ; Pick up rec buf count
0602 1545 MOVB #CDB_C_ACPT,CDB_B_STA(R5) ; Change state to "accept"
0602 1546 SETBIT #CDB_V_ACPT,CDB_W_STS(R5) ; Indicate ACCEPT pending
0602 1547 ACCEPT ; ACCEPT the connection
0602 1548 MSGADR = W^MSG_FORK,- ; Message address
0602 1549 DGADR = W^DG_FORK,- ; Psuedo interrupt rtn
0602 1550 ERRADR = W^CONN_ERR,- ; Error address
0602 1551 INITCR = #1,- ; Allow for messages
0602 1552 INITDG = R0,- ; Receive buffers
0602 1553 CONDAT = CONN_DATA,- ; Accept data
0602 1554 AUXSTR = (R5) ; Auxiliary structure (CDB)
0602 1555
0602 1556
0602 1557
0602 1558
0602 1559
0602 1560
0602 1561
0602 1562
0602 1563
0602 1564
0602 1565
0602 1566
0602 1567
0602 1568
0602 1569
0602 1570
0602 1571
0602 1572
0602 1573
0602 1574
0602 1575
0602 1576
0602 1577
0602 1578
0602 1579
0602 1580
0602 1581
0602 1582
0602 1583
0602 1584
0602 1585
0602 1586
0602 1587
0602 1588
0602 1589
0602 1590
0602 1591
0602 1592
0602 1593
0602 1594
0602 1595
0602 1596
0602 1597
0602 1598
0602 1599
0602 1600
0602 1601
0602 1602
0602 1603
0602 1604
0602 1605
0602 1606
0602 1607
0602 1608
0602 1609
0602 1610
0602 1611
0602 1612
0602 1613
0602 1614
0602 1615
0602 1616
0602 1617
0602 1618
0602 1619
0602 1620
0602 1621
0602 1622
0602 1623
0602 1624
0602 1625
0602 1626
0602 1627
0602 1628
0602 1629
0602 1630
0602 1631
0602 1632
0602 1633
0602 1634
0602 1635
0602 1636
0602 1637
0602 1638
0602 1639
0602 1640
0602 1641
0602 1642
0602 1643
0602 1644
0602 1645
0602 1646
0602 1647
0602 1648
0602 1649
0602 1650
0602 1651
0602 1652
0602 1653
0602 1654
0602 1655
0602 1656
0602 1657
0602 1658
0602 1659
0602 1660
0602 1661
0602 1662
0602 1663
0602 1664
0602 1665
0602 1666
0602 1667
0602 1668
0602 1669
0602 1670
0602 1671
0602 1672
0602 1673
0602 1674
0602 1675
0602 1676
0602 1677
0602 1678
0602 1679
0602 1680
0602 1681
0602 1682
0602 1683
0602 1684
0602 1685
0602 1686
0602 1687
0602 1688
0602 1689
0602 1690
0602 1691
0602 1692
0602 1693
0602 1694
0602 1695
0602 1696
0602 1697
0602 1698
0602 1699
0602 1700
0602 1701
0602 1702
0602 1703
0602 1704
0602 1705
0602 1706
0602 1707
0602 1708
0602 1709
0602 1710
0602 1711
0602 1712
0602 1713
0602 1714
0602 1715
0602 1716
0602 1717
0602 1718
0602 1719
0602 1720
0602 1721
0602 1722
0602 1723
0602 1724
0602 1725
0602 1726
0602 1727
0602 1728
0602 1729
0602 1730
0602 1731
0602 1732
0602 1733
0602 1734
0602 1735
0602 1736
0602 1737
0602 1738
0602 1739
0602 1740
0602 1741
0602 1742
0602 1743
0602 1744
0602 1745
0602 1746
0602 1747
0602 1748
0602 1749
0602 1750
0602 1751
0602 1752
0602 1753
0602 1754
0602 1755
0602 1756
0602 1757
0602 1758
0602 1759
0602 1760
0602 1761
0602 1762
0602 1763
0602 1764
0602 1765
0602 1766
0602 1767
0602 1768
0602 1769
0602 1770
0602 1771
0602 1772
0602 1773
0602 1774
0602 1775
0602 1776
0602 1777
0602 1778
0602 1779
0602 1780
0602 1781
0602 1782
0602 1783
0602 1784
0602 1785
0602 1786
0602 1787
0602 1788
0602 1789
0602 1790
0602 1791
0602 1792
0602 1793
0602 1794
0602 1795
0602 1796
0602 1797
0602 1798
0602 1799
0602 1800
0602 1801
0602 1802
0602 1803
0602 1804
0602 1805
0602 1806
0602 1807
0602 1808
0602 1809
0602 1810
0602 1811
0602 1812
0602 1813
0602 1814
0602 1815
0602 1816
0602 1817
0602 1818
0602 1819
0602 1820
0602 1821
0602 1822
0602 1823
0602 1824
0602 1825
0602 1826
0602 1827
0602 1828
0602 1829
0602 1830
0602 1831
0602 1832
0602 1833
0602 1834
0602 1835
0602 1836
0602 1837
0602 1838
0602 1839
0602 1840
0602 1841
0602 1842
0602 1843
0602 1844
0602 1845
0602 1846
0602 1847
0602 1848
0602 1849
0602 1850
0602 1851
0602 1852
0602 1853
0602 1854
0602 1855
0602 1856
0602 1857
0602 1858
0602 1859
0602 1860
0602 1861
0602 1862
0602 1863
0602 1864
0602 1865
0602 1866
0602 1867
0602 1868
0602 1869
0602 1870
0602 1871
0602 1872
0602 1873
0602 1874
0602 1875
0602 1876
0602 1877
0602 1878
0602 1879
0602 1880
0602 1881
0602 1882
0602 1883
0602 1884
0602 1885
0602 1886
0602 1887
0602 1888
0602 1889
0602 1890
0602 1891
0602 1892
0602 1893
0602 1894
0602 1895
0602 1896
0602 1897
0602 1898
0602 1899
0602 1900
0602 1901
0602 1902
0602 1903
0602 1904
0602 1905
0602 1906
0602 1907
0602 1908
0602 1909
0602 1910
0602 1911
0602 1912
0602 1913
0602 1914
0602 1915
0602 1916
0602 1917
0602 1918
0602 1919
0602 1920
0602 1921
0602 1922
0602 1923
0602 1924
0602 1925
0602 1926
0602 1927
0602 1928
0602 1929
0602 1930
0602 1931
0602 1932
0602 1933
0602 1934
0602 1935
0602 1936
0602 1937
0602 1938
0602 1939
0602 1940
0602 1941
0602 1942
0602 1943
0602 1944
0602 1945
0602 1946
0602 1947
0602 1948
0602 1949
0602 1950
0602 1951
0602 1952
0602 1953
0602 1954
0602 1955
0602 1956
0602 1957
0602 1958
0602 1959
0602 1960
0602 1961
0602 1962
0602 1963
0602 1964
0602 1965
0602 1966
0602 1967
0602 1968
0602 1969
0602 1970
0602 1971
0602 1972
0602 1973
0602 1974
0602 1975
0602 1976
0602 1977
0602 1978
0602 1979
0602 1980
0602 1981
0602 1982
0602 1983
0602 1984
0602 1985
0602 1986
0602 1987
0602 1988
0602 1989
0602 1990
0602 1991
0602 1992
0602 1993
0602 1994
0602 1995
0602 1996
0602 1997
0602 1998
0602 1999
0602 2000
0602 2001
0602 2002
0602 2003
0602 2004
0602 2005
0602 2006
0602 2007
0602 2008
0602 2009
0602 2010
0602 2011
0602 2012
0602 2013
0602 2014
0602 2015
0602 2016
0602 2017
0602 2018
0602 2019
0602 2020
0602 2021
0602 2022
0602 2023
0602 2024
0602 2025
0602 2026
0602 2027
0602 2028
0602 2029
0602 2030
0602 2031
0602 2032
0602 2033
0602 2034
0602 2035
0602 2036
0602 2037
0602 2038
0602 2039
0602 2040
0602 2041
0602 2042
0602 2043
0602 2044
0602 2045
0602 2046
0602 2047
0602 2048
0602 2049
0602 2050
0602 2051
0602 2052
0602 2053
0602 2054
0602 2055
0602 2056
0602 2057
0602 2058
0602 2059
0602 2060
0602 2061
0602 2062
0602 2063
0602 2064
0602 2065
0602 2066
0602 2067
0602 2068
0602 2069
0602 2070
0602 2071
0602 2072
0602 2073
0602 2074
0602 2075
0602 2076
0602 2077
0602 2078
0602 2079
0602 2080
0602 2081
0602 2082
0602 2083
0602 2084
0602 2085
0602 2086
0602 2087
0602 2088
0602 2089
0602 2090
0602 2091
0602 2092
0602 2093
0602 2094
0602 2095
0602 2096
0602 2097
0602 2098
0602 2099
0602 2100
0602 2101
0602 2102
0602 2103
0602 2104
0602 2105
0602 2106
0602 2107
0602 2108
0602 2109
0602 2110
0602 2111
0602 2112
0602 2113
0602 2114
0602 2115
0602 2116
0602 2117
0602 2118
0602 2119
0602 2120
0602 2121
0602 2122
0602 2123
0602 2124
0602 2125
0602 2126
0602 2127
0602 2128
0602 2129
0602 2130
0602 2131
0602 2132
0602 2133
0602 2134
0602 2135
0602 2136
0602 2137
0602 2138
0602 2139
0602 2140
0602 2141
0602 2142
0602 2143
0602 2144
0602 2145
0602 2146
0602 2147
0602 2148
0602 2149
0602 2150
0602 2151
0602 2152
0602 2153
0602 2154
0602 2155
0602 2156
0602 2157
0602 2158
0602 2159
0602 2160
0602 2161
0602 2162
0602 2163
0602 2164
0602 2165
0602 2166
0602 2167
0602 2168
0602 2169
0602 2170
0602 2171
0602 2172
0602 2173
0602 2174
0602 2175
0602 2176
0602 2177
0602 2178
0602 2179
0602 2180
0602 2181
0602 2182
0602 2183
0602 2184
0602 2185
0602 2186
0602 2187
0602 2188
0602 2189
0602 2190
0602 2191
0602 2192
0602 2193
0602 2194
0602 2195
0602 2196
0602 2197
0602 2198
0602 2199
0602 2200
0602 2201
0602 2202
0602 2203
0602 2204
0602 2205
0602 2206
0602 2207
0602 2208
0602 2209
0602 2210
0602 2211
0602 2212
0602 2213
0602 2214
0602 2215
0602 2216
0602 2217
0602 2218
0602 2219
0602 2220
0602 2221
0602 2222
0602 2223
0602 2224
0602 2225
0602 2226
0602 2227
0602 2228
0602 2229
0602 2230
0602 2231
0602 2232
0602 2233
0602 2234
0602 2235
0602 2236
0602 2237
0602 2238
0602 2239
0602 2240
0602 2241
0602 2242
0602 2243
0602 2244
0602 2245
0602 2246
0602 2247
0602 2248
0602 2249
0602 2250
0602 2251
0602 2252
0602 2253
0602 2254
0602 2255
0602 2256
0602 2257
0602 2258
0602 2259
0602 2260
0602 2261
0602 2262
0602 2263
0602 2264
0602 2265
0602 2266
0602 2267
0602 2268
0602 2269
0602 2270
0602 2271
0602 2272
0602 2273
0602 2274
0602 2275
0602 2276
0602 2277
0602 2278
0602 2279
0602 2280
0602 2281
0602 2282
0602 2283
0602 2284
0602 2285
0602 2286
0602 2287
0602 2288
0602 2289
0602 2290
0602 2291
0602 2292
0602 2293
0602 2294
0602 2295
0602 2296
0602 2297
0602 2298
0602 2299
0602 2300
0602 2301
0602 2302
0602 2303
0602 2304
0602 2305
0602 2306
0602 2307
0602 2308
0602 2309
0602 2310
0602 2311
0602 2312
0602 2313
0602 2314
0602 2315
0602 2316
0602 2317
0602 2318
0602 2319
0602 2320
0602 2321
0602 2322
0602 2323
0602 2324
0602 2325
0602 2326
0602 2327
0602 2328
0602 2329
0602 2330
0602 2331
0602 2332
0602 2333
0602 2334
0602 2335
0602 2336
0602 2337
0602 2338
0602 2339
0602 2340
0602 2341
0602 2342
0602 2343
0602 2344
0602 2345
0602 2346
0602 2347
0602 2348
0602 2349
0602 2350
0602 2351
0602 2352
0602 2353
0602 2354
0602 2355
0602 2356
0602 2357
0602 2358
0602 2359
0602 2360
0602 2361
0602 2362
0602 2363
0602 2364
0602 2365
0602 2366
0602 2367
0602 2368
0602 2369
0602 2370
0602 2371
0602 2372
0602 2373
0602 2374
0602 2375
0602 2376
0602 2377
0602 2378
0602 2379
0602 2380
0602 2381
0602 2382
0602 2383
0602 2384
0602 2385
0602 2386
0602 2387
0602 2388
0602 2389
0602 2390
0602 2391
0602 2392
0602 2393
0602 2394
0602 2395
0602 2396
0602 2397
0602 2398
0602 2399
0602 2400
0602 2401
0602 2402
0602 2403
0602 2404
0602 2405
0602 2406
0602 2407
0602 2408
0602 2409
0602 2410
0602 2411
0602 2412
0602 2413
0602 2414
0602 2415
0602 2416
0602 2417
0602 2418
0602 2419
0602 2420
0602 2421
0602 2422
0602 2423
0602 2424
0602 2425
0602 2426
0602 2427
0602 2428
0602 2429
0602 2430
0602 2431
0602 2432
0602 2433
0602 2434
0602 2435
0602 2436
0602 2437
0602 2438
0602 2439
0602 2440
0602 2441
0602 2442
0602 2443
0602 2444
0602 2445
0602 2446
0602 2447
0602 2448
0602 2449
0602 2450
0602 2451
0602 2452
0602 2453
0602 2454
0602 2455
0602 2456
0602 2457
0602 2458
0602 2459
0602 2460
0602 2461
0602 2462
0602 2463
0602 2464
0602 2465
0602 2466
0602 2467
0602 2468
0602 2469
0602 2470
0602 2471
0602 2472
0602 2473
0602 2474
0602 2475
0602 2476
0602 2477
0602 2478
0602 2479
0602 2480
0602 2481
0602 2482
0602 2483
0602 2484
0602 2485
0602 2486
0602 2487
0602 2488
0602 2489
0602 2490
0602 2491
0602 2492
0602 2493
0602 2494
0602 2495
0602 2496
0602 2497
0602 2498
0602 2499
0602 2500
0602 2501
0602 2502
0602 2503
0602 2504
0602 2505
0602 2506
0602 2507
0602 2508
0602 2509
0602 2510
0602 2511
0602 2512
0602 2513
0602 2514
0602 2515
0602 2516
0602 2517
0602 2518
0602 2519
0602 2520
0602 2521
0602 2522
0602 2523
0602 2524
0602 2525
0602 2526
0602 2527
0602 2528
0602 2529
0602 2530
0602 2531
0602 2532
0602 2533
0602 2534
0602 2535
0602 2536
0602 2537
0602 2538
0602 2539
0602 2540
0602 2541
0602 2542
0602 2543
0602 2544
0602 2545
0602 2546
0602 2547
0602 2548
0602 2549
0602 2550
0602 2551
0602 2552
0602 2553
0602 2554
0602 2555
0602 2556
0602 2557
0602 2558
0602 2559
0602 2560
0602 2561
0602 2562
0602 2563
0602 2564
0602 2565
0602 2566
0602 2567
0602 2568
0602 2569
0602 2570
0602 2571
0602 2572
0602 2573
0602 2574
0602 2575
0602 2576
0602 2577
0602 2578
0602 2579
0602 2580
0602 2581
0602 2582
0602 2583
0602 2584
0602 2585
0602 2586
0602 2587
0602 2588
0602 2589
0602 2590
0602 2591
0602 2592
0602 2593
0602 2594
0602 2595
0602 2596
0602 2597
0602 2598
0602 2599
0602 2600
0602 2601
0602 2602
0602 2603
0602 2604
0602 2605
0602 2606
0602 2607
0602 2608
0602 2609
0602 2610
0602 2611
0602 2612
0602 2613
0602 2614
0602 2615
0602 2616
0602 2617
0602 2618
0602 2619
0602 2620
0602 2621
0602 2622
0602 2623
0602 2624
0602 2625
0602 2626
0602 2627
0602 2628
0602 2629
0602 2630
0602 2631
0602 2632
0602 2633
0602 2634
0602 2635
0602 2636
0602 2637
0602 2638
0602 2639
0602 2640
0602 2641
0602 2642
0602 2643
0602 2644
0602 2645
0602 2646
0602 2647
0602 2648
0602 2649
0602 2650
0602 2651
0602 2652
0602 2653
0602 2654
0602 2655
0602 2656
0602 2657
0602 2658
0602 2659
0602 2660
0602 2661
0602 2662
0602 2663
0602 2664
0602 2665
0602 2666
0602 2667
0602 2668
0602 2669
0602 2670
0602 2671
0602 26
```



```
0693 1588
0693 1589
0693 1590 : CONNECT (or ACCEPT) succeeded
0693 1591
0693 1592 : If no status bits are set then enter the "run" state and complete the
0693 1593 : pending IOSM_STARTUP request. If any status bits are set -- which can
0693 1594 : happen if we are talking to ourselves since we do both an ACCEPT and a
0693 1595 : CONNECT in that case -- then wait.
0693 1596
0693 1597 CONN_FIN:
54 A5 53 D0 0693 1598 MOVL R3,CDB_L_CDT(R5) : Set ptr to CDT
0697 1599 CONN_FIN1:
54 55 D0 0697 1600 MOVL R5,R4 : Copy CDT address
5B A4 94 069A 1601 CLRB CDB_B_RSTCNT(R4) : Init failed restart counter
34 A4 D4 069D 1602 CLRL CDB_L_ABSTIME(R4) : Don't inhibit DISCONNECT
55 50 A4 D0 06A0 1603 MOVL CDB_L_UCB(R4),R5 : Restore UCB pointer
64 A5 20 AA 06A4 1604 BICW #UCBSM_POWER,UCBSW_STS(R5) : Any powerfail recovery is done
3A A4 B5 06AB 1605 TSTW CDB_W_STS(R4) : All quiet?
1D 12 06AB 1606 BNEQ 40$ : If NEQ no, wait
53 30 A4 D0 06AD 1607 MOVL CDB_L_SETMODE(R4),R3 : Get SETMODE IRP
18 13 06B1 1608 BEQL 50$ : If EQL then none
30 A4 D4 06B3 1609 CLRL CDB_L_SETMODE(R4) : Detach IRP from CDB
10 20 A3 06 06B6 1610 BBC #IOSV_STARTUP,IRPSW_FUNC(R3),50$ : If BC then wrong IRP
3F A4 01 90 06B8 1611 MOVW #CDB_C_OPEN,CDB_B_STA(R4) : Update current state
50 01 3C 06C4 1612 SETBIT #CDB_V_RUN,CDB_W_STS(R4) : Allow data message traffic
021A 30 06C7 1614 BSBW SUC_TRB_IOPOST : Setup status
05 06CA 1615 40$: RSB : Post IRP with "success"
06CB 1616
06CB 1617 50$: BUG_CHECK INCONSTATE,FATAL
06CF 1618
06CF 1619 CHECK_REMOTE: : Check remote connect data
06CF 1620
06CF 1621 :
06CF 1622 : 0-15(R2) Contain our process name (who remote is connecting to)
06CF 1623 : 16-31(R2) Contain remote's process name
06CF 1624 : 32-47(R2) Contain connect data
06CF 1625
06CF 1626 PUSHF #M<R0,R2,R3,R4> : Save some registers
06D1 1627
06D1 1628 MOVL R2,R4 : Make stable msg pointer
54 52 D0 06D1 1629 MOVW #OLD_C_PROT,CDB_W_REMPROT(R5) : Assume remote is old protocol
20 A4 5B A5 00 B0 06D4 1629 CMPC3 #PROC_C_NAM,PROC_NAM,32(R4) : Check the connect data
FA13 CF 06 29 06D8 1630 BEQL 10$ : If EQL then old style
05 13 06DF 1631 MOVW 32(R4),CDB_W_REMPROT(R5) : Pickup version + system id's
5B A5 20 A4 B0 06E1 1632 CMPC3 #PROC_C_NAM,PROC_NAM,16(R4) : Check the connect proc nam
10 A4 FA05 CF 06 29 06E6 1633 10$:
06ED 1634 POPR #M<R0,R2,R3,R4> : Restore regs (but save CC's)
1D BA 06ED 1635 RSB : Return condition codes
05 06EF 1636
06F0 1637
06F0 1638
06F0 1639
06F0 1640 : Error after connection established - VC disconnect most likely.
06F0 1641
06F0 1642 : If the CDT is the UCBSL_TWIN_CDT then simply do a DISCONNECT. This CDT is
06F0 1643 : used for receives on connects to ourselves. SCS will call us again for the
06F0 1644 : other half of that connection with the local CDB's CDT -- at that time, as
```



```

06F0 1645 : in all other cases, we will run-down the CDB.
06F0 1646
06F0 1647 : Inputs:
06F0 1648 : R0 = Status
06F0 1649 : R3 --> CDT
06F0 1650 : R4 --> PDT
06F0 1651
06F0 1652 : CONN_ERR:
54 5C A3 D0 06F0 1653 : MOVL CDT$L_AUXSTRUC(R3),R4 : Pick up associated CDB
55 50 A4 D0 06F4 1654 : MOVL CDB_L_UCB(R4),R5 : Pick up UCB address
0094 C5 53 D1 06F8 1655 : CMPL R3,UCB$L_TWIN_CDT(R5) : Is this the "local receive" CDT ?
: 6F 12 06FD 1656 : BNEQ ZAP CDB : If NEQ no, ZAP the CDB
: 0094 C5 D4 06FF 1657 : CLRL UCB$L_TWIN_CDT(R5) : Else, detach it from the UCB
54 10 A3 D0 0703 1658 : MOVL CDT$L_PDT(R3),R4 : Recover the PDT
: 0707 1659 : DISCONNECT : Tell SCS to cleanup.
: 070D 1660 : RSB : Done
: 070E 1661

```

```
070E 1663      .SBTTL CANCEL, Cancel I/O routine
070E 1664
070E 1665      :++
070E 1666      CANCEL, Cancels an I/O operation in progress
070E 1667      :
070E 1668      :
070E 1669      This routine cancels all I/O on the tributary.
070E 1670      :
070E 1671      Inputs:      R2 = channel number
070E 1672                  R3 = current IRP address
070E 1673                  R4 = PCB address
070E 1674                  R5 = UCB address
070E 1675                  RB = Cancel reason code: 0 => $CANCEL; 1 => $DASSGN
070E 1676
070E 1677                  IPL = FIPL
070E 1678      :
070E 1679      Outputs:     R0-R3 are destroyed.
070E 1680      :
070E 1681      :
070E 1682      :--
070E 1683      CANCEL:
0210 8F      BB 070E 1684      PUSHR      #^M<R4,R9>      : Cancel an I/O operation
50 52      D0 0712 1685      : Save registers
025A 30 0712 1686      MOVL      R2,R0      : Copy channel number
10 50      E9 0715 1687      BSBW      XLATE CHAN      : Translate channel
01 58      D1 0718 1688      BLBC      R0,20$      : Br if none
09 12      D1 071B 1689      CMPL      R8,#1      : $DASSGN ?
50 3E A9 9A 071E 1690      BNEQ      10$      : If NEQ then no
00E0 C540 B4 0720 1691      MOVZBL   CDB_B_TRB_ADDR(R9),R0 : Pick up trib address
40 10      B4 0724 1692      CLRW      UCBSW-VEC-CHAN(R5)[R0] : Zero channel entry
0729 1693 10$:      BSBW      ZAP_CDB_R9      : Clear all CDB I/O
072B 1694
0210 8F      BA 072B 1695 20$:      POPR      #^M<R4,R9>      : Restore registers
5C A5      B5 072F 1696      TSTW      UCBSW REFC(R5)      : Last reference to unit?
01 13      13 0732 1697      BEQL      CAN_DEV      : If EQL yes, shutdown the device
05 05      05 0734 1698      RSB      : Return to caller
0735 1699
```

```
0735 1701 .SBTTL CAN_DEV, Device shutdown routine
0735 1702
0735 1703 :++
0735 1704 CAN_DEV - Device shutdown routine
0735 1705
0735 1706
0735 1707 This routine is called to shutdown the CI device. All tributaries are
0735 1708 zapped so that they will eventually run-down and be deleted.
0735 1709
0735 1710 Inputs: R3 = IRP address
0735 1711 R5 = UCB address
0735 1712
0735 1713 IPL = FIPL
0735 1714
0735 1715 Outputs: R0-R2 are clobbered.
0735 1716
0735 1717
0735 1718 --
0735 1719 CAN_DEV:
0735 1720 BBCC #UCBSV_CN_INITED,- ; Shutdown the device
0737 1721 UCB$W_DEVSTS(R5),50$ ; Br if dev not initd
073A 1722 PUSHR #^M<R3,R4,R5> ;
073C 1723
073C 1724 : Zap each tributary
073C 1725
073C 1726 MOVL #MAX_TRB-1,R3 ; Loop counter (zero indexed)
54 00A0 C543 D0 073F 1727 20$: MOVL UCB$C_VEC_CDB(R5)[R3],R4 ; Get next CDB
0745 1728 BEQL 30$ ; Br if none
0747 1729 BSBB ZAP_CDB ; Cancel all I/O on trib
F3 53 F4 0749 1730 30$: SOBGEQ R3,20$ ; Loop
074C 1731
074C 1732 : Remove our listener
074C 1733
53 0090 C5 D0 074C 1734 MOVL UCB$L_LIS_CDT(R5),R3 ; Pick up listening CDT
0751 1735 BEQL 40$ ; None
0753 1736 CLRL UCB$L_LIS_CDT(R5) ; and clear any trace
54 0084 C5 D0 0757 1737 MOVL UCB$L_PDT(R5),R4 ; PDT address, just in case
075C 1738 DISCONNECT ; Clear our name out of table
0762 1739 40$:
0762 1740 : Clean up the UCB
0762 1741
FFCF 8F AA 0762 1742 BICW2 #^C<UCBSM_ONLINE!UCBSM_POWER>,- ; Reset status
64 A5 0766 1743 UCB$W_STSR5) ;
0768 1744
38 BA 0768 1745 POPR #^M<R3,R4,R5> ; Restore registers
05 076A 1746 50$: RSB ; Return
```

```
076B 1748 .SBTTL ZAP_CDB, Shutdown the tributary
076B 1749
076B 1750 :++
076B 1751 ZAP_CDB - Shutdown the tributary
076B 1752 ZAP_CDB_R9 - Shutdown the tributary, get CDB address from R9
076B 1753
076B 1754
076B 1755 This routine is called to abort all I/O pending for this tributary.
076B 1756
076B 1757 1) Disconnect the Virtual Circuit
076B 1758 2) Cancel all outstanding I/O, abort all IRP, deallocate rcv'd buffers.
076B 1759 3) Idle the CDB.
076B 1760
076B 1761 Inputs: R9 = CDB address (ZAP_CDB_R9 only, else not used)
076B 1762 R5 = UCB address
076B 1763 R4 = CDB address (ZAP_CDB only, else garbage)
076B 1764
076B 1765 IPL = FIPL
076B 1766
076B 1767 Outputs: R0-R1 are destroyed.
076B 1768
076B 1769 :--
54 59 D0 076B 1770 ZAP_CDB_R9: MOVL R9,R4 ; Setup proper CDB pointer
076E 1771 ZAP_CDB:
076E 1772
076E 1773 : If a DISCONNECT is issued on a connection that already has a
076E 1774 DISCONNECT pending, SCS thinks that something is wrong the port
076E 1775 and disconnects all circuits using it. Therefore, make sure
076E 1776 we do not issue a second DISCONNECT for at least 10 seconds after
076E 1777 the last one was issued. That should be enough time for normally
076E 1778 functioning circuits to complete a DISCONNECT dialogue. If the
076E 1779 DISCONNECT is still pending after 10 seconds, its probably okay
076E 1780 to try it again in order to allow the user to run-down all I/O on
076E 1781 this channel.
076E 1782
076E 1783 TSTL CDB_L_CDT(R4) ; Any CDT connected ?
0771 1784 BEQL 2$ ; If EQL, no DISCONNECT needed
0773 1785 SUBL3 CDB_L_ABSTIME(R4),- ; Get seconds since last
0776 1786 G^EXE$GL_ABSTIM,R0 DISCONNECT
077C 1787 CMPL R0,#10 ; At least 10 seconds?
077F 1788 BLSSU 3$ ; If LSSU can't DISCONNECT
0781 1789
0781 1790 2$: PUSHR #M<R2,R3,R4,R5> ; Save regs
0783 1791 BSBB 5$ ; Use subr call so that SCS's
0785 1792 DISCONNECT code can return to
0785 1793 a caller's caller
0785 1794 POPR #M<R2,R3,R4,R5> ; Restore regs
0787 1795 3$: RSB ; Done
0788 1796
0788 1797 5$:
0788 1798
0788 1799
0788 1800 DISCONNECT may return to our caller before returning here since SCS
0788 1801 has to enter into a dialogue with the remote node. Therefore, the
0788 1802 stack must be clear.
0788 1803
0788 1804 FORK immediately after returning from the DISCONNECT in order to
; make sure SCS will return all Xmt IRPs it knows about before we
```


				0788	1805	:	return the ones that are left. This is not usually necessary since			
				0788	1806	:	in most cases SCS will queue the DISCONNECT completion to the end of			
				0788	1807	:	its own fork queue after all Xmt completion notifications have been			
				0788	1808	:	queued.			
				0788	1809	:				
				0788	1810	:	If CDB_V_DISC is already set, then there is a DISCONNECT or FORK			
				0788	1811	:	already in progress. Do the DISCONNECT again, in case SCS is stuck,			
				0788	1812	:	but simply return since the previous DISCONNECT will complete and			
				0788	1813	:	the processing will continue from there.			
				0788	1814	:				
				0788	1815	:				
	55	54	D0	078D	1816	CLRBIT	#CDB_V_RUN,CDB_W_STS(R4)	No longer in RUN state		
	53	54	A5	D0	0790	MOVL	R4,R5	Save CDB pointer over call		
		06	12	0794	1817	MOVL	CDB_L_CDT(R5),R3	Pick up CDT address		
	25	3A	A5	03	E3	BNEQ	10\$	If NEQ then CDT was there		
				05	0796	BBCS	#CDB_V_DISC,CDB_W_STS(R5),30\$	FORK to continue		
					079B	RSB		Return if already FORKING		
					079C					
	54	10	A3	D0	079C	1821				
	34	A5	00000000'GF	D0	07A0	1822	10\$: MOVL	CDT\$L_PDT(R3),R4	Pick up PDT address	
	0A	3A	A5	03	E3	07A0	1823	MOVL	G^EXE\$GL_ABSTIM,CDB_L_ABSTIME(R5)	Save DISCONNECT start time
					05	07A8	1824	BBCS	#CDB_V_DISC,CDB_W_STS(R5),20\$	Show we are disconnecting
						07AD	1825	DISCONNECT	#0	Tell SCS to do it again
						07B6	1826	RSB		Done
						07B7	1827			
						07B7	1828	20\$: DISCONNECT	#0	Do it
	00000000'GF		16	07C0	1829	30\$: JSB	G^EXE\$FORK			FORK to synchronize cleanup
	54	55	D0	07C6	1830	MOVL	R5,R4			Recover CDB address
	55	50	A4	D0	07C9	1831	MOVL	CDB_L_UCB(R4),R5		Recover UCB address
						07CD	1832	SETIPL	UCB\$B_FIPL(R5)	Sync with UCB
						07D1	1833	CLRBIT	#CDB_V_DISC,CDB_W_STS(R4)	Show we are back
		04	10	07D6	1834	BSBB	40\$			Finish processing
						07D8	1835	SETIPL	#CDB_C_FIPL	Restore IPL
				05	07DB	1836	RSB			
						07DC	1837			
						07DC	1838	40\$:		
						07DC	1839	:	Complete pending IO\$_SETMODE, if any	
						07DC	1840	:		
	54	A4	D4	07DC	1841	CLRL	CDB_L_CDT(R4)	:	Get rid of any trace	
	53	30	A4	D0	07DF	1842	MOVL	CDB_L_SETMODE(R4),R3	Recover IRP	
		0E	13	07E3	1843	BEQL	50\$:	None there	
		30	A4	D4	07E5	1844	CLRL	CDB_L_SETMODE(R4)	Remove it from the CDB	
	50	01	7D	07E8	1845	MOVQ	S^#55\$ NORMAL,R0	Assume IO\$V_SHUTDOWN		
	09	20	A3	07	E1	07E8	1846	BBC	#IO\$V_SHUTDOWN,IRP\$W_FUNC(R3),60\$	If BC then IO\$V_STARTUP
		0105	30	07F0	1847	BSBW	IOPOST	:	Send IRP to IOPOST	
						07F3	1848			
						07F3	1849	50\$:		
						07F3	1850	:	Complete all Receive IRP's	
						07F3	1851	:		
	53	20	B4	0F	07F3	1852	REMQUE	@CDB_Q_RCV_IRP(R4),R3	Get next RCV IRP	
		05	1D	07F7	1853	BVS	70\$:	If VS then none	
		00E3	30	07F9	1854	BSBW	ABORT_IRP_POST	:	Abort the I/O request	
		F5	11	07FC	1855	BRB	50\$:	Get next entry	
						07FE	1856			
						07FE	1857	70\$:		
						07FE	1858	:	Deallocate all Receive buffers	
						07FE	1859	:		
	50	28	B4	0F	07FE	1860	REMQUE	@CDB_Q_RCV_MSG(R4),R0	Get next buffer	
		04	1D	0802	1861	BVS	80\$:	If VS then empty	

	3F	10	0804	1862	BSBB	DEALLMEM		
	F6	11	0806	1863	BRB	70\$: Get rid of it
			0808	1864				: Get next entry
			0808	1865	80\$:			
			0808	1866				
			0808	1867				
			0808	1868				
53	18	84	0F	0808	1868	REMQUE	@CDB_Q_XMT_IRP(R4),R3	: Get next IRP
		05	1D	080C	1869	BVS	90\$: If VS then none
	00	CE	30	080E	1870	BSBW	ABORT_IRP_POST	: Abort the I/O request
		F5	11	0811	1871	BRB	80\$: Loop
				0813	1872			
				0813	1873	90\$:		
				0813	1874			
				0813	1875			
3F	A4	00	90	0813	1876	MOVB	#CDB_C_IDLE,CDB_B_STA(R4)	: Reinit CDB state
			05	0817	1877	RSB		
				0818	1878			

```

0818 1880 .SBTTL MSG_FORK, Fork process for receipt of Seq Messages
0818 1881
0818 1882 :++
0818 1883 :MSG_FORK - Process received MSG
0818 1884 :
0818 1885 :Inputs:
0818 1886 :
0818 1887 :R1 = Bytes send/received
0818 1888 :R2 --> Start of user data
0818 1889 :R3 --> CDT
0818 1890 :R4 --> PDT
0818 1891 :
0818 1892 :IPL = FIPL
0818 1893 :
0818 1894 :Outputs:
0818 1895 :
0818 1896 :--
0818 1897 MSG_FORK:
05 0818 1898 DEALLOC_MSG_BUF_REG ; Deallocate the message buffer
0818 1899 RSB
081C 1900

```

```
.SBTTL DG_FORK, Fork process for receipt of DG
081C 1902
081C 1903
081C 1904
081C 1905
081C 1906
081C 1907
081C 1908
081C 1909
081C 1910
081C 1911
081C 1912
081C 1913
081C 1914
081C 1915
081C 1916
081C 1917
081C 1918
081C 1919
081C 1920
081C 1921
081C 1922
081C 1923
0820 1924
0822 1925
0826 1926
0829 1927
082D 1928
082F 1929
0832 1930
0832 1931
0834 1932
0838 1933
083A 1934
083C 1935
083E 1936
083E 1937
0842 1938
0842 1939
0845 1940
0845 1941
084B 1942
084B 1943
084B 1944
084B 1945
084B 1946
084B 1947
084B 1948
084B 1949
084F 1950
0852 1951
0855 1952
0858 1953
085A 1954
085D 1955
0860 1956
0862 1957
0866 1958

54 5C A3 D0
53 52 01 C3
55 52 20 C2
55 0B A2 32
55 DF 1B
52 55 C0
63 FF 8F 91
11 12
51 D7
F4 14
50 52 D0
00000000'GF 17

DG_FORK:
    MOVL CDTSL_AUXSTRUC(R3),R4
    BEQL EMPTY
    SUBL3 #1,R2,R3
    SUBL #32,R2
    CVTWL 8(R2),R5
    BGEQ 20$
    ADDL R5,R2
    10$: INCL R3
    CMPB #-1,(R3)
    BNEQ DG
    DECL R1
    BGTR 10$
    20$: BUG_CHECK INCONSTATE,FATAL
    EMPTY: MOVL R2,R0
    DEALLMEM: JMP G^CON$DRVDEALMEM
    DG:
        : Update counters
        :
        ASSUME CDB_L_BSN EQ 4+CDB_L_BRC
        ASSUME CDB_L_DBR EQ 4+CDB_L_BSN
        ASSUME CDB_L_DBS EQ 4+CDB_L_DBR
        MOVAB CDB_L_BRC(R4),R5
        BLBC R0,5$
        ADDL #4,R5
        5$: ADDL R1,(R5)
        BCC 10$
        MNEGL #1,(R5)
        10$: INCL 8(R5)
        BCC 20$
        MNEGL #1,8(R5)
        20$: MOVL CDB_L_UCB(R4),R5

: Pick up pointer to CDB
: Closed CDB, discard
: Make a biased copy of msg ptr
: Reset R2 to head of PPD buffer
: Get offset to CXB header
: If GEQ then bug
: Reset R2 to head of CXB buffer
:
: Advance to next byte
: Pad byte ?
: If NEQ not pad byte
: Reduce count
: If LEQ then no data
:
: Illegal offset
:
: Pick up buffer
:
: Deallocate buffer
:
: Point to receive counter base
: If LBC, then rcv
: Advance to xmt counter base
: Update byte count
: Br if no overflow
: Else, latch it
: Update message count
: If CC, no overflow
: Else, latch it
: Pick up ptr to UCB
```



```
1D 50 E8 086A 1959 BLBS R0,SEND_FORK ; IF LBS then xmt complete
086D 1960
086D 1961 : RECEIVE complete - if there is a pending receive I/O request,
086D 1962 : complete it. Otherwise, queue the buffer.
086D 1963
086D 1964 ASSUME CDB_V_RUN EQ 0
086D 1965 BLBC CDB_W_STS(R4),EMPTY ; Br if trib not in RUN state
OE A2 D1 3A A4 E9 086D 1965 DECB CDB_B_RCV_FQ(R4) ; Dec the buffer count
3D A4 97 0871 1966 SUBW3 R2,R3,CXB$W_OFFSET(R2) ; Store offset to message
53 52 A3 0874 1967 MOVW R1,CXB$W_LENGTH(R2) ; Set size
OC A2 51 B0 0879 1968 REMQUE @CDB_Q_RCV_IRP(R4),R3 ; Remove waiting IRP
53 20 B4 OF 087D 1969 BVC FINISH_RCV_IO ; If VC then gone one, finish
2C 2C 1C 0881 1970 : the I/O & exit
0883 1971 : Queue receive msg for late
2C B4 62 OE 0883 1972 : Fill the receive buffer pool
0086 31 0887 1973
088A 1974 SEND_FORK:
088A 1975 : TRANSMIT completed. Locate and deque XMIT IRP and post it.
088A 1976 :
088A 1977 : NOTE: the IRP's may be returned out of sequence on a power fail.
088A 1978 :
088A 1979 :
50 51 10 9C 088A 1981 ROTL #16,R1,R0 ; Size in R0 high word
50 01 B0 088E 1982 MOVW #SS$ NORMAL,R0 ; Status in low word
51 18 A4 9E 0891 1983 MOVAB CDB_Q_XMT_IRP(R4),R1 ; Address queue header
53 51 D0 0895 1984 20$: MOVL R1,R3 ; Make a copy
53 63 D0 0898 1985 MOVL (R3),R3 ; Get next IRP
51 53 D1 089B 1986 CMPL R3,R1 ; Back to head of queue?
2C A3 08 13 089E 1987 BEQL 50$ ; If EQL then yes, bugcheck
53 52 D1 08A0 1988 CMPL R2,IRP$L_SVAPTE(R3) ; Buffer address match?
F2 12 08A4 1989 BNEQ 20$ ; If NEQ no, try again
53 63 OF 08A6 1990 REMQUE (R3),R3 ; Remove IRP from queue
39 11 08A9 1991 BRB SUC_IRB_IOPOST ; Complete the I/O with trib
08AB 1992 : info stuffed into IOSI2
08AB 1993
08AB 1994 50$: BUG_CHECK INCONSTATE,FATAL
08AF 1995
```

```
.SBTTL FINISH_RCV_IO, Finish receive I/O processing

:++
FINISH_RCV_IO - Finish receive I/O processing

This routine finishes receive processing and sends the IRP back to IOPOST.
The receive free list is filled and a receive is started if needed.

Inputs:      R2 = message buffer address
             R3 = IRP address
             R4 = CDB address
             R5 = UCB address

IPL = FIPL

Outputs:     R0-R4 are clobbered. All other registers are preserved.

--
FINISH_RCV_IO:
    MOVL R2,IRPSL_SVAPTE(R3)      : Finish receive I/O request
    MOVZWL CXBSW_OFFSET(R2),(R2)  : Save block address
    ADDL R2,(R2)                  : Store offset to message
    MOVL IRPSL_IOST2(R3),4(R2)    : Make it a pointer
    MOVW CXBSW_LENGTH(R2),R1      : Set address of user buffer
    CMPW R1,IRPSW_BCNT(R3)        : Get size of transfer
    BGTRU 10$                    : Request larger than actual?
    MOVW R1,IRPSW_BCNT(R3)        : Br GTRU then yes
    MOVL IRPSW_BCNT-2(R3),R0      : Set size to transfer
    MOVW #SS$_NORMAL,R0          : Setup size of xfer in high word
    BNEQ SUC_TRB_IOPOST          : Setup status in low word
    MOVZWL #SS$_CTLERR,R0        : Br if success
    CLRL R1                      : Set data path error
    BRB IOPOST                   : Init second longword
    : Post it

    50 2C 7D 08DF 2034 ABORT_IRP_POST:
    14 11 08E2 2035 MOVQ S*SS$_ABORT,R0 : Setup IOSB image
    08E4 2036 BRB IOPOST : Finish up
    08E4 2037

    51 00002800 8F D0 08E4 2038 SUC_TRB_IOPOST:
    3D A4 06 91 08EB 2040 MOVL #XMSM_STS_ACTIVE!- : Successful Trib I/O completion
    07 18 08EB 2041 XMSM_STS_RUNNING,R1 : Set device dependent bits to indicate
    51 00001000 8F C8 08EF 2042 #RBFTR,CDB_B_RCV_FQ(R4) : that the circuit is running
    38 A3 50 7D 08F1 2043 IOPOST BLEQU IOPOST : Receive queue under threshold?
    00000000'GF 17 08F8 2044 BLSL #XMSM_STS_BUFFAIL,R1 : If LEQU then no
    08FC 2045 IOPOST: MOVQ R0,IRPSL_IOST1(R3) : Signal buffer threshold problems
    0902 2046 JMP G*COM$POST : Store IOSB image
    : Post IRP
```

```
0902 2048 .SBTTL FILLRCVLIST, Fill receive buffer list
0902 2049 .SBTTL ADDRCLVLIST, Move IRP buffer to free list
0902 2050 :++
0902 2051 FILLRCVLIST - Add to the receive buffer list
0902 2052 ADDRCLVLIST - Add IRP buffer to free list
0902 2053
0902 2054 This routine is entered to make sure that the receive buffer pool is full.
0902 2055 If it is not, buffers are allocated and queued to the list until it is.
0902 2056
0902 2057 For ADDRCLVLIST, any buffer attached to the IRP is added to the free list
0902 2058 even if the list is already filled.
0902 2059
0902 2060 Inputs: R3 - IRP address (ADDRCLVLIST only)
0902 2061 R4 - CDB address
0902 2062 R5 - UCB address
0902 2063
0902 2064 Outputs: Only R0-R2 are clobbered.
0902 2065
0902 2066 :--
0902 2067 .ENABL LSB
0902 2068 ADDRCLVLIST:
52 2C A3 D0 0902 2069 MOVL IRP$L_SVAPTE(R3),R2 : Add IRP buffer to free list
      08 13 0906 2070 BEQL FILLRCVLIST : Get buffer, if any
      2C A3 D4 0908 2071 CLRL IRP$L_SVAPTE(R3) : If none, fill rcv list if needed
      1E 11 090B 2072 PUSHQ R3 : Detach the buffer
      090E 2073 BRB 20$ : Save regs
      0910 2074 : Add buffer to free list
      0910 2075 FILLRCVLIST:
      0910 2076 PUSHQ R3 : Save regs
51 3C A4 3D A4 91 0913 2077 10$: CMPB CDB_B_RCV_FQ(R4),CDB_B_RCV_CNT(R4) : Should new block be added?
      45 1E 0918 2078 BGEQU 50$ : If GEQU no - list filled
      004C 8F A1 091A 2079 ADDW3 #CXB$C_OVERHEAD,CDB_W_BUFSIZ(R4),R1 : Compute block size need
      00000000 GF 16 0921 2080 JSB G*EXE$ALONONPAGED : Allocate nonpaged memory
      35 50 E9 0927 2081 BLBC R0,50$ : If LBC then failure
      08 A2 51 B0 092A 2082 MOVW R1,IRP$W_SIZE(R2) : Insert block size
      092E 2083 20$: :
      092E 2084 : Give SCS receive datagram
      092E 2085
      092E 2086 MOVL CDB_L_CDT(R4),R3 : Pick up CDT address
20 A3 53 54 A4 D0 0932 2087 CMPB UCB$B_CN_PORT(R5),CDT$B_RSTATION(R3) : Talking to ourselves?
      009E C5 91 0938 2088 BNEQ 30$ : If NEQ, no
      05 12 093A 2089 MOVL UCB$L_TWIN_CDT(R5),R3 : Yes, use other CDT
      0094 C5 D0 093F 2090 30$: MOVL UCB$L_PDT(R5),R4 : and PDT address
      0084 C5 D0 0944 2091 MOVZBW S*#DYN$C_CXB,IRP$B_TYPE(R2) : Insert block type
      0A A2 1B 9B 0948 2092 QUEUE_DG BUF : Put the block on the free que
      09 50 E9 094B 2093 BLBC R0,40$ : Br if failure
      04 AE D0 094E 2094 MOVL 4(SP),R4 : Pick up CDB pointer
      3D A4 96 0952 2095 INCB CDB_B_RCV_FQ(R4) : Bump free que count
      BC 11 0955 2096 BRB 10$ : Try for more
      50 52 D0 0957 2097 40$: MOVL R2,R0 : Pick up the buffer
      03 13 095A 2098 BEQL 50$ : There is none
      FEE6 30 095C 2099 BSBW DEALLMEM : Get rid of the buffer
      095F 2100
      095F 2101 50$: POPQ R3 : Restore regs
      0962 2102 RSB : Return
      0963 2103
      0963 2104 .DSABL LSB
```

```
0963 2106 .SBTTL XLATE, Translate Channel to CDB address
0963 2107
0963 2108 :++
0963 2109 : XLATE - Translate Channel to CDB address
0963 2110 :
0963 2111 : This routine is called to return the CDB address for a particular
0963 2112 : channel.
0963 2113 :
0963 2114 :
0963 2115 : Inputs:      R3 = IRP address
0963 2116 :             R5 = UCB address
0963 2117 :
0963 2118 : Outputs:     R0 - status return for success of call.
0963 2119 :
0963 2120 :             R9 = CDB address if successful
0963 2121 :                 1 otherwise
0963 2122 :
0963 2123 :             R1,R2 are clobbered, all other registers are preserved.
0963 2124 :
0963 2125 :--
0963 2126 XLATE:
0963 2127     MOVZWL IRP$W_CHAN(R3),R0      ; Translate CHAN into CDB address
0963 2128     BSBB   XLATE_CHAN            ; Get channel
0963 2129     MOVB   R1,IRP$B_INDEX(R3)    ; Translate channel
0963 2130     MOVL   R9,IRP$L_CDB(R3)      ; Save index in IRP
0963 2131     RSB    R0,R0                ; Store CDB address in IRP
0963 2132     RSB    R0,R0                ; Return to caller
0963 2133 XLATE_CHAN:
0963 2134     MOVZBL #MAX_TRB-1,R1        ; Setup loop counter (zero indexed)
0963 2135 10$:    CMPW   R0,UCB$W_VEC_CHAN(R5)[R1] ; Channels match?
0963 2136     BEQL   40$                ; Br if yes - got it
0963 2137     SOBGEQ R1,10$              ; Loop
0963 2138 30$:    MOVZWL #SS$_DEVINACT,R0      ; Return channel offline
0963 2139     MOVL   #1,R9               ; Setup 'R9 invalid' flag
0963 2140     RSB    R0,R0                ; And leave
0963 2141
0963 2142 40$:    :
0963 2143     : Found match on channel
0963 2144     :
0963 2145     MOVL   UCB$L_VEC_CDB(R5)[R1],R9 ; Get CDB address
0963 2146     BEQL   50$                ; Br if no CDB address - error
0963 2147     MOVZWL S^#SS$_NORMAL,R0    ; Set successful return status
0963 2148     RSB    R0,R0                ; Return
0963 2149
0963 2150 50$:    BUG_CHECK INCONSTATE,FATAL
0963 2151
```



```
0999 2153 .SBTTL VALIDATE_P2, Validate P2 buffer parameters
0999 2154
0999 2155 ++
0999 2156 VALIDATE_P2 - Validate P2 buffer parameters
0999 2157
0999 2158 This routine is called to validate the P2 buffer parameters. The parameters
0999 2159 are checked against a parameter table which verifies that the minimum value
0999 2160 and maximum value is not violated, and that status flags are set or clear
0999 2161 as required.
0999 2162
0999 2163 The way in this routine is written, the require word of the verification
0999 2164 table can only have 1 bit set at a time.
0999 2165
0999 2166 Inputs: R1 = Address of parameter verification table
0999 2167 R2 = Status word from UCB or CDB
0999 2168 R3 = IRP address
0999 2169 R5 = UCB address
0999 2170 R9 = If low bit clear then ptr to context block (CDB or UCB)
0999 2171 If low bit set then no context block exists
0999 2172
0999 2173 IPL = FIPL or ASTDEL
0999 2174
0999 2175 Outputs: R0 = status return of parameters
0999 2176 R1 = i.d. of parameter causing problem on error
0999 2177
0999 2178 All other registers are preserved.
0999 2179
0999 2180 --
0999 2181 VALIDATE_P2:
0999 2182 PUSHRR #^M<R1,R3,R5,R6,R7,R8> : Validate P2 buffer parameters
0999 2183 : Save registers
0999 2184 : NB: R1 must be on top of stack
0999 2185 MOVLL @IRPSL_SVAPTE(R3),R6 : Get system P2 buffer address
0999 2186 MOVZWL IRPSW_BCNT(R3),R8 : Get size of P2 buffer
0999 2187 DIVL #6,R8 : Get number of params in P2
0999 2188 BRB 40$ : Treat as none if too few bytes
0999 2189 10$:
0999 2190 : Loop to check next parameter in P2 buffer
0999 2191 MOVZWL (R6)+,R0 : Get parameter type from P2
0999 2192 MOVL (R6)+,R5 : Get parameter value from P2
0999 2193 MOVL (SP),R7 : Get parameter table address
0999 2194 20$:
0999 2195 : Loop to check P2 buffer parameter to circuit parameter table
0999 2196 BICW3 #^C<PRM_M_TYPE>,(R7)+,R1 : Get next param i.d.
0999 2197 BEQL 50$ : If EQL, at end of table
0999 2198 CMPW R0,R1 : Parameters match?
0999 2199 BEQL 30$ : Br if yes
0999 2200 ADDL2 #PARAM_C_ENTRY-2,R7 : Else, skip to next parameter
0999 2201 BRB 20$ : Try next parameter
0999 2202 30$:
0999 2203 : Match found - check min,max,valid,invalid
0999 2204 MOVZWL (R7)+,R3 : Get offset/width
0999 2205 BLBS R9,33$ : If LBS then no current block
0999 2206 EXTZV #OFF_V_VALUE,#OFF_S_VALUE,R3,R0 : Get offset
0999 2207 EXTZV #OFF_V_WIDTH,#OFF_S_WIDTH,R3,R3 : Get width
0999 2208
0999 2209
```

01EA	8F	BB	0999	2182
56	2C	B3	DO	099D
58	32	A3	3C	09A1
	58	06	C6	09A5
		4F	11	09A8
				09AA
				09AA
				09AA
50	86	3C	09AA	2191
55	86	DO	09AD	2192
57	6E	DO	09B0	2193
			09B3	2194
			09B3	2195
			09B3	2196
51	87	F000	8F	AB
			46	13
	51	50	B1	09B8
		05	13	09BE
	57	0A	C0	09C0
		EE	11	09C3
				09C5
				09C5
				09C5
				09C5
	53	87	3C	09C5
		15	EB	09C8
50	53	0A	00	EF
53	53	06	0A	EF
				09D0

```
53 6940 53 00 EF 09D5 2210 EXTZV #0,R3,(R9)[R0],R3 : Get current value
53 55 B1 09DB 2211 CMPW R5,R3 : Value change ?
53 19 13 09DE 2212 BEQL 40$ : If EQL no, try next param
87 55 B1 09E0 2213 33$: CMPW R5,(R7)+ : Is the value too small?
87 1C 1F 09E3 2214 BLSSU 50$ : Br if yes - error
87 55 B1 09E5 2215 CMPW R5,(R7)+ : Is the value too big?
87 17 1A 09E8 2216 BGTRU 50$ : Br if yes - error
55 87 B0 09EA 2217 MOVW (R7)+,R5 : Pick up required
55 05 13 09ED 2218 BEQL 35$ : None
52 55 B3 09EF 2219 BITW R5,R2 : Check required bit
52 0D 13 09F2 2220 BEQL 50$ : Br if not on - error
52 87 B3 09F4 2221 35$: BITW (R7)+,R2 : Check invalid bits
52 08 12 09F7 2222 BNEQ 50$ : Br if on - error
AE 58 F5 09F9 2223 40$: SOBGTR R8,10$ : Br if more parameters
50 01 3C 09FC 2224 MOVZWL S^#SS$ _NORMAL,R0 : Set success return
50 06 11 09FF 2225 BRB 60$ : And return
6E 51 3C 0A01 2226 :
50 14 D0 0A04 2227 50$: MOVZWL R1,(SP) : Return bad parameter type
01EA 8F BA 0A07 2228 MOVL #55$ _BADPARAM,R0 : Set error return
05 0A0B 2229 60$: POPR #*M<R1,R3,R5,R6,R7,R8> : Restore registers
2230 RSB : Return to caller
```

```

.OAOC 2232      .SBTTL UNPACK_P2_BUF, Unpack a P2 parameter from P2 buffer
.OAOC 2233
.OAOC 2234      :++
.OAOC 2235      UNPACK_P2_BUF - Unpack a P2 parameter from P2 buffer
.OAOC 2236
.OAOC 2237      :
.OAOC 2238      This routine is called to get a P2 parameter from the P2 buffer.
.OAOC 2239
.OAOC 2240      Inputs:      R1 = Parameter type code
.OAOC 2241                  R3 = IRP address
.OAOC 2242                  R5 = UCB address
.OAOC 2243
.OAOC 2244                  IPL = IPL$_ASTDEL to allow user paging.
.OAOC 2245
.OAOC 2246      Outputs:    R0 = $$$_NORMAL if successful
.OAOC 2247                  $$$_INSFARG otherwise
.OAOC 2248                  R2 = Parameter value if success else destroyed
.OAOC 2249
.OAOC 2250                  All other registers are preserved.
.OAOC 2251
.OAOC 2252      :--
.OAOC 2253      UNPACK_P2_BUF:
.OAOC 2254      POSHR    #^M<R5,R6,R7>      : Unpack P2 buffer
.OAOC 2255      MOVL     @IRP$L_SVAPTE(R3),R6 : Save registers
.OAOC 2256      MOVZWL   IRP$L_BCNT(R3),R7   : Get system P2 buffer address
.OAOC 2257      DIVL     #6,R7                 : Get size of P2 buffer
.OAOC 2258      BEQL     20$,R7               : Get number of params in P2
.OAOC 2259      MOVZWL   S^$_NORMAL,R0       : Treat as none if too few bytes
.OAOC 2260      10$:                          : Assume success
.OAOC 2261      :
.OAOC 2262      : Loop to check next parameter in P2 buffer
.OAOC 2263      MOVZWL   (R6)+,R5              : Get parameter type from P2
.OAOC 2264      MOVL     (R6)+,R2              : Get parameter value from P2
.OAOC 2265      CMPW     R1,R5                : Parameters match?
.OAOC 2266      BEQL     30$,R5              : Br if yes
.OAOC 2267      SOBGTR  R7,10$              : Br if more parameters
.OAOC 2268      OA2E 2268
.OAOC 2269      20$: MOVZWL   $$$_INSFARG,R0 : Return error
.OAOC 2270      30$: POPR     #^M<R5,R6,R7> : Restore registers
.OAOC 2271      RSB                      : Return to caller

```

CNDRIVER
V04-000

- VAX/VMS DECnet-CI Class Driver
CN_END, End of driver

B 14

16-SEP-1984 01:19:27
5-SEP-1984 00:11:06

VAX/VMS Macro V04-00
[DRIVER.SRC]CNDRIVER.MAR;1

Page 53
(39)

```
0A38 2273 .SBTTL CN_END, End of driver
0A38 2274
00000A40 0A38 2275 . = <.+15>&<-16>
0A40 2276
00000018 0A40 2277 PATCH:: .LONG 32-8
00000A48 0A44 2278 .LONG PATCH+8
00000A60 0A48 2279 .BLKB 32-8
0A60 2280
0A60 2281 :++
0A60 2282 : Label that marks the end of the driver
0A60 2283 :--
0A60 2284
0A60 2285 CN_END: ; Last location in driver
0A60 2286
0A60 2287 .END
```


CNDRIVER
Symbol table

- VAX/VMS DECnet-CI Class Driver

C 14

16-SEP-1984 01:19:27 VAX/VMS Macro V04-00
5-SEP-1984 00:11:06 [DRIVER.SRC]CNDRIVER.MAR;1

Page 54
(39)

```

$$$ = 00000020 R 02
$$$NUM = 00000000
$$$OFF = 0000004C
$$$TYP = 000063F3
$$OP = 00000002
ABORT_IRP_POST = 000008DF R 03
ABORT_REQ = 000002B8 R 03
ABORT_REQ1 = 000002B8 R 03
ABORT_START = 00000226 R 03
ACCEPT = 00000626 R 03
ADDRCVLIST = 00000902 R 03
ALT_START = 000001B8 R 03
AT$NULL = 00000005
BUG$INCONSTATE = ***** X 03
CANCEL = 0000070E R 03
CAN_DEV = 00000735 R 03
CDB_B_DUMMY = 0000005A
CDB_B_FIPL = 0000000B
CDB_B_RCV_CNT = 0000003C
CDB_B_RCV_FQ = 0000003D
CDB_B_REMSYS = 00000059
CDB_B_REMVER = 00000058
CDB_B_RSTCNT = 0000005B
CDB_B_STA = 0000003F
CDB_B_TRB_ADDR = 0000003E
CDB_B_TYPE = 0000000A
CDB_C_ACPT = 00000004
CDB_C_CONN = 00000002
CDB_C_FIPL = 00000006
CDB_C_IDLE = 00000000
CDB_C_LENGTH = 00000060
CDB_C_LSTN = 00000003
CDB_C_OPEN = 00000001
CDB_L_ABSTIME = 00000034
CDB_L_BRC = 00000040
CDB_L_BSN = 00000044
CDB_L_CDT = 00000054
CDB_L_DBR = 00000048
CDB_L_DBS = 0000004C
CDB_L_FPC = 0000000C
CDB_L_FR3 = 00000010
CDB_L_FR4 = 00000014
CDB_L_SETMODE = 00000030
CDB_L_UCB = 00000050
CDB_M_RUN = 00000001
CDB_Q_FORK = 00000000
CDB_Q_RCV_IRP = 00000020
CDB_Q_RCV_MSG = 00000028
CDB_Q_XMT_IRP = 00000018
CDB_V_ACPT = 00000002
CDB_V_CONN = 00000001
CDB_V_DISC = 00000003
CDB_V_REJECT = 00000004
CDB_V_RUN = 00000000
CDB_W_BUFSIZ = 00000038
CDB_W_REMPROT = 00000058
CDB_W_SIZE = 00000008

```

```

CDB_W_STS = 0000003A
CDT$B-RSTATION = 00000020
CDT$B-AUXSTRUC = 0000005C
CDT$B-PBT = 00000010
CHECK-REMOTE = 000006CF R 03
CLR_IRP = 00000115 R 03
CN$DDT = 00000000 R 03
CN_END = 00000A60 R 03
CN-FUNCTABLE = 00000038 R 03
COM$DRVDEALMEM = ***** X 03
COM$POST = ***** X 03
CONN = 00000591 R 03
CONN_ABO = 000005E4 R 03
CONN_DATA = 00000100 R 03
CONN_ERR = 000006F0 R 03
CONN_FIN = 00000693 R 03
CONN_FIN1 = 00000697 R 03
COUNT_C_ENTRY = 00000004
CRE$B_INTD = 00000024
CXB$C_HEADER = 00000048
CXB$C_OVERHEAD = 0000004C
CXB$W_LENGTH = 0000000C
CXB$W_OFFSET = 0000000E
DC$SCOM = 00000020
DDB$B_DDT = 0000000C
DEALLMEM = 00000845 R 03
DEV$M_IDV = 04000000
DEV$M_NET = 00002000
DEV$M_ODV = 08000000
DEV$M_REC = 00000001
DG = 00000848 R 03
DG_FORK = 0000081C R 03
DPT$C_LENGTH = 00000038
DPT$C_VERSION = 00000004
DPT$INITAB = 00000038 R 02
DPT$M_SCS = 00000008
DPT$REINITAB = 00000062 R 02
DPT$TAB = 00000000 R 02
DYN$C_CIDG = 00000038
DYN$C_CRB = 00000005
DYN$C_CXB = 00000018
DYN$C_DDB = 00000006
DYN$C_DPT = 0000001E
DYN$C_NET = 00000017
DYN$C_ORB = 00000049
DYN$C_UCB = 00000010
EMPTY = 00000842 R 03
EXESABORTIO = ***** X 03
EXESALONONPAGED = ***** X 03
EXES$BUFFRQUOTA = ***** X 03
EXES$FINISHIO = ***** X 03
EXES$FORK = ***** X 03
EXES$GL_ABSTIM = ***** X 03
EXES$QI$RETURN = ***** X 03
EXES$READCHK = ***** X 03
EXES$WRITECHK = ***** X 03
FILLRCVLIST = 00000910 R 03

```

CNDRIVER
Symbol table

- VAX/VMS DECnet-CI Class Driver

D 14

16-SEP-1984 01:19:27 VAX/VMS Macro V04-00
5-SEP-1984 00:11:06 [DRIVER.SRC]CNDRIVER.MAR;1

Page 55
(39)

FINISH_ERR	000002C4	R	03	NMASC_CTCIR_DBR	= 000003F2		
FINISH_RCV_IO	000008AF	R	03	NMASC_CTCIR_DBS	= 000003F3		
FINISH_REQ	000002D5	R	03	NMASC_DPX_FOL	= 00000000		
FINISH_SUC	000002C1	R	03	NMASC_DPX_HAL	= 00000001		
FUNCTAB_LEN	= 0000004C			NMASC_LINCN_LOO	= 00000001		
GET_BUF	0000012D	R	03	NMASC_LINCN_NOR	= 00000000		
GET_CHAR_RBUF	000004D9	R	03	NMASC_PCCI_MRB	= 00000479		
GET_CHAR_WBUF	000004DD	R	03	NMASC_PCCI_MST	= 00000AFA		
IOSV_CLR_COUNT	= 0000000A			NMASC_PCCI_TRI	= 00000474		
IOSV_CTRL	= 00000009			NMASC_PCLI_BFN	= 00000451		
IOSV_RD_COUNT	= 00000008			NMASC_PCLI_BUS	= 00000AF1		
IOSV_SHUTDOWN	= 00000007			NMASC_PCLI_CON	= 00000456		
IOSV_STARTUP	= 00000006			NMASC_PCLI_DUP	= 00000457		
IOS_READBLK	= 00000021			NMASC_STATE_OFF	= 00000001		
IOS_SENSEMODE	= 00000027			NMASC_STATE_ON	= 00000000		
IOS_SETCHAR	= 0000001A			NMASC_CNT_COU	= 00008000		
IOS_SETMODE	= 00000023			NMASC_CNT_WID	= 0000000D		
IOS_VIRTUAL	= 0000003F			OFF_M_VALUE	= 000003FF		
IOS_WRITEBLK	= 00000020			OFF_S_VALUE	= 0000000A		
IOCSMNTVER	*****	X	03	OFF_S_WIDTH	= 00000006		
IOCSRETURN	*****	X	03	OFF_V_VALUE	= 00000000		
IOPOST	000008F8	R	03	OFF_V_WIDTH	= 0000000A		
IRPSB_INDEX	00000040			OLD_C_PROT	= 00000000		
IRPSB_RMOD	= 0000000B			ORBSB_FLAGS	= 0000000B		
IRPSB_TYPE	= 0000000A			ORBSL_OWNER	= 00000000		
IRPSL_BCNT	= 00000032			ORBSM_PROT_16	= 00000001		
IRPSL_CDB	00000054			ORBSW_PROT	= 00000018		
IRPSL_EXTEND	= 00000054			P1	= 00000000		
IRPSL_IOST1	= 00000038			P2	= 00000004		
IRPSL_IOST2	= 0000003C			PARAM_C_ENTRY	= 0000000C		
IRPSL_SEGVBN	= 00000048			PATCH	00000A40	RG	03
IRPSL_SVAPTE	= 0000002C			PBSL_PDT	= 0000002C		
IRPSM_FUNC	= 00000002			PCBSL_JIB	= 00000080		
IRPSQ_NT_PRVMSK	= 00000040			PDTSL_DEALRMSG	= 00000024		
IRPSV_FUNC	= 00000001			PDTSL_QUEUEDG	= 0000003C		
IRPSW_BCNT	= 00000032			PDTSL_REJECT	= 0000004C		
IRPSW_BOFF	= 00000030			PDTSL_SENDRGDG	= 0000007C		
IRPSW_CHAN	= 00000028			PR\$ IPL	= 00000012		
IRPSW_FUNC	= 00000020			PRM_M_INVALID	= 00008000		
IRPSW_SIZE	= 00000008			PRM_M_MAX	= 00002000		
IRPSW_STS	= 0000002A			PRM_M_MIN	= 00001000		
JIBSL_BYTCNT	= 00000020			PRM_M_REQUIRE	= 00004000		
LINE_CNT_NUM	= 00000000			PRM_M_TYPE	= 00000FFF		
LINE_CNT_TABLE	= 000000EE	R	03	PROC_C_NAM	= 00000006		
LINE_PRH_NUM	= 00000004			PROC_NAM	000000F0	R	03
LINE_PRH_TABLE	= 000000AA	R	03	QIORET	000002B2	R	03
LISTEN	000003AF	R	03	RBFMAX	= 0000001F		
LIS_ERR	0000041F	R	03	RBFMIN	= 00000009		
LIS_FORK	00000602	R	03	RBFTHR	= 00000006		
MASKH	= 00000080			RCV_FDT	00000184	R	03
MASKL	= 00000000			RCV_START	00000229	R	03
MAX_TRB	= 00000010			REJECT	0000061D	R	03
MSG_FORK	00000818	R	03	SBSL_PBCONNX	= 00000014		
NEW_CDB	0000030A	R	03	SBCSB_RSTATION1	= 0000003C		
NEW_TRIB	000002DB	R	03	SBO\$C_LENGTH	= 00000050		
NMASC_CTCIR_BRC	= 000003E8			SBO_LNG	= 00000070		
NMASC_CTCIR_BSN	= 000003E9			SCSB\$ACCEPT	*****	GX	03

CNDRIVER
Symbol table

- VAX/VMS DECnet-CI Class Driver E 14

16-SEP-1984 01:19:27 VAX/VMS Macro V04-00
5-SEP-1984 00:11:06 [DRIVER.SRC]CNDRIVER.MAR;1

Page 56
(39)

SCSS\$CONFIG SYS	*****	X	03	VALIDATE_P2	00000999	R	03
SCSS\$CONNECT	*****	X	03	VECSL_UNITINIT	= 00000018		
SCSS\$DISCONNECT	*****	X	03	XLATE-	00000963	R	03
SCSS\$GB_SYSTEMID	*****	X	03	XLATE_CHAN	00000972	R	03
SCSS\$GW_MAXDG	*****	X	02	XMSM_STS_ACTIVE	= 00000800		
SCSS\$LISTEN	*****	GX	03	XMSM_STS_BUFFAIL	= 00001000		
SEND_FORK	0000088A	R	03	XMSM_STS_RUNNING	= 00002000		
SENSEMODE_FDT	0000042E	R	03	XMT_FDT	00000125	R	03
SENSE_C_BUF	= 00000080			XMT_RCV_FDT_CO	00000190	R	03
SENSE_TABLE	00000426	R	03	XMT_START	000001D2	R	03
SETMODE_CTRL	0000035E	R	03	ZAP_CDB	0000076E	R	03
SETMODE_FDT	0000023D	R	03	ZAP_CDB_R9	0000076B	R	03
SIZ...	= 00000001						
SS\$ABORT	= 0000002C						
SS\$ACCVIO	= 0000000C						
SS\$BADPARAM	= 00000014						
SS\$BUFFEROVF	= 00000601						
SS\$CTRLERR	= 00000054						
SS\$DEVACTIVE	= 000002C4						
SS\$DEVALRALLOC	= 00000641						
SS\$DEVINACT	= 000020D4						
SS\$DEVOFFLINE	= 00000084						
SS\$INSFARG	= 00000114						
SS\$NORMAL	= 00000001						
SS\$REMRSRC	= 0000206C						
START TRIB	0000052E	R	03				
SUC TRIB IOPOST	000008E4	R	03				
TRIB_CNT_NUM	= 00000004						
TRIB_CNT_TABLE	= 000000DC	R	03				
TRIB_PRM_NUM	= 00000003						
TRIB_PRM_TABLE	= 00000084	R	03				
UCBS\$CN_PORT	0000009E						
UCBS\$DEVCLASS	= 00000040						
UCBS\$DIPL	= 0000005E						
UCBS\$FIPL	= 0000000B						
UCBS\$RCV_CNT	0000009F						
UCBS\$CN_LENGTH	= 00000100						
UCBS\$LENGTH	= 00000090						
UCBS\$DEVCHAR	= 00000038						
UCBS\$DGHDRSZ	00000098						
UCBS\$LIS_CDT	00000090						
UCBS\$PDT	= 00000084						
UCBS\$TWIN_CDT	00000094						
UCBS\$VEC_CDB	000000A0						
UCBSM_CN_INITED	= 00000001						
UCBSM_ONLINE	= 00000010						
UCBSM_POWER	= 00000020						
UCBSV_CN_INITED	= 00000000						
UCBSV_POWER	= 00000005						
UCBSW_DEVBUSIZ	= 00000042						
UCBSW_DEVSTS	= 00000068						
UCBSW_DUMMY	0000009C						
UCBSW_REFC	= 0000005C						
UCBSW_STS	= 00000064						
UCBSW_VEC_CHAN	000000E0						
UNIT_INIT	00000110	R	03				
UNPACK_P2_BUF	00000A0C	R	03				

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000100 (256.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$105_PROLOGUE	0000006D (109.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	00000A60 (2656.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:00.85
Command processing	113	00:00:00.39	00:00:04.55
Pass 1	811	00:00:26.09	00:01:43.79
Symbol table sort	0	00:00:03.73	00:00:16.44
Pass 2	417	00:00:05.69	00:00:23.01
Symbol table output	15	00:00:00.20	00:00:00.82
Psect synopsis output	0	00:00:00.02	00:00:00.06
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1387	00:00:36.20	00:02:29.55

The working set limit was 2400 pages.
217048 bytes (424 pages) of virtual memory were used to buffer the intermediate code.
There were 200 pages of symbol table space allocated to hold 3593 non-local and 117 local symbols.
2287 source lines were read in Pass 1, producing 22 object records in Pass 2.
68 pages of virtual memory were used to define 62 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	36
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	50

3866 GETS were required to define 50 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:CNDRIVER/OBJ=OBJ\$:CNDRIVER MSRC\$:CNDRIVER/UPDATE=(ENH\$:CNDRIVER)+EXECMLS/LIB

0107 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0107	0108	0109	0110	0111	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------